51CTO.com
技术博客Blog

博客月刊

blog.51cto.com

2014年05月
总第 06 期

- Powershell检测AD账户密码过期时间并邮件通知
- 在python下比celery更加简单的异步任务队列RQ
- IT项目中存储设备的选型
- 用VMware Fusion来“穿越”
- 插件式的80后程序员是怎样在夹缝中求生存?

目录

Powershell 检测 AD 账户密码过期时间并邮件通知

作者：李晓松 地址：http://lixiaosong.blog.51cto.com/705126/1409113

我记得在坛子里流传这一份用 PS1.0 版本实现此功能的脚本本来想直接使用，但居然发现不会用呵呵。 后来一想直接写一个得了，此脚本主要实现了两个功能 ： 一能判断账户密码的过期时间并通过邮件通知到账户，二是将这些即将过期的账户信息累计通知到管理员。

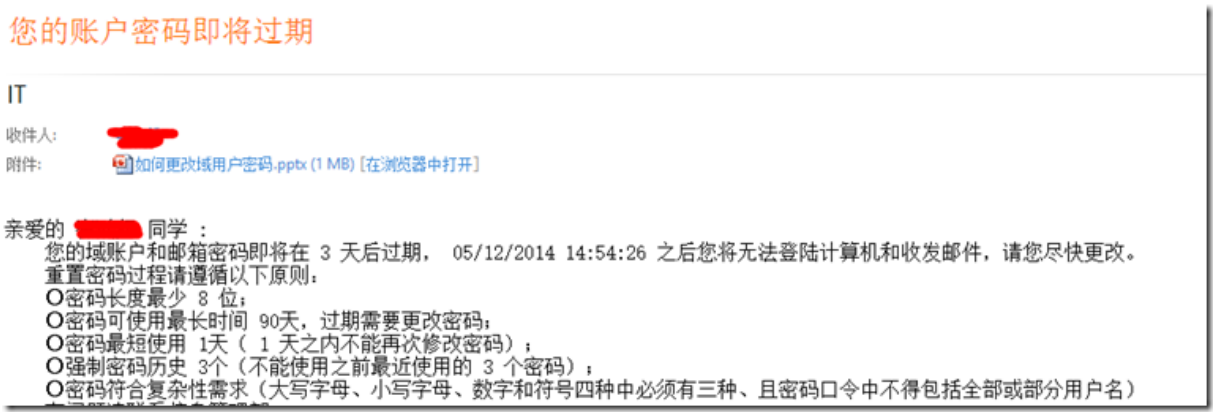
```
##############################################
# Author:Lixiaosong
# Email:lixs@ourgame.com;lixiaosong8706@gmail.com
# For:检测 AD 密码过期时间并邮件通知
#Version:1.0
##################################################
Import-Module Activedirectory
$alladuser=get-aduser -searchbase "OU=IT,DC=contoso,DC=com" -filter *
| %{$_.Samaccountname}
$userlist = @()
#####################################################
#检测 AD 密码过期时间并邮件通知相应账户
#####################################################
foreach ($user in $alladuser){
#密码最后一次更改时间
$pwdlastset=Get-ADUser $user -Properties * | %{$_.passwordlastset}
```

```
#密码的过期时间

$pwdlastday=($pwdlastset).adddays(90)

#当前时间

$now=get-date

#判断账户是否设置了永不过期

$neverexpire=get-aduser $user -Properties * |%{$_.PasswordNeverExpires}

#距离密码过期的时间

$expire_days=($pwdlastday - $now).Days

#判断过期时间天小于 15 天的并且没有设置密码永不过期的账户

if($expire_days -lt 15 -and $neverexpire -like "false" ){

$chineseusername= Get-ADUser $user   -Properties * | %{$_.Displayname}

#邮件正文

$Emailbody=

"亲爱的 $chineseusername 同学：

您的域账户和邮箱密码即将在 $expire_days 天后过期， $pwdlastday 之后您将无法登陆计算机和收

发邮件，请您尽快更改。

重置密码过程请遵循以下原则：

○密码长度最少 8 位；

○密码可使用最长时间 90 天，过期需要更改密码；

○密码最短使用 1 天（ 1 天之内不能再次修改密码）；

○强制密码历史 3 个（不能使用之前最近使用的 3 个密码）；

○密码符合复杂性需求（大写字母、小写字母、数字和符号四种中必须有三种、且密码口令中不得包括全

部或部分用户名）
```

```
"

Send-MailMessage -from "it@contoso.com" -to "$user@contoso.com" -subject "您的账户密码即
将过期" -body $Emailbody -Attachments D:\script\如何更改域用户密码.pptx -smtpserver
mail.contoso.com -Encoding ([System.Text.Encoding]::UTF8)

##############################################

#查找账户的密码过期时间并发送至管理员账户

##############################################

$username=Get-ADUser $user    -Properties *

$userobject=New-object psobject

$userobject | Add-Member -membertype noteproperty -Name 用户名                -value
$username.displayname

$userobject | Add-Member -membertype noteproperty -Name 邮箱                  -Value
$username.mail

$userobject | Add-Member -membertype noteproperty -Name 最后一次密码设置    -Value
$username.Passwordlastset

$userobject | Add-Member -membertype noteproperty -Name 密码过期时间        -Value
$pwdlastday

$userobject | Add-Member -membertype noteproperty -Name 距离密码过期天数    -Value
$expire_days

$userlist+=$userobject

}

}

$EmailbodyHTML=$userlist|
```

```
sort-object 距离密码过期天数 |

ConvertTo-Html |

Out-String

Send-Mailmessage -from "it@contoso.com" –to “itmanager@contoso” -Bodyashtml

$EmailbodyHTML -Subject "管理员通知" -smtpserver mail.contoso.com -Encoding

([System.Text.Encoding]::UTF8)
```

实现的结果：



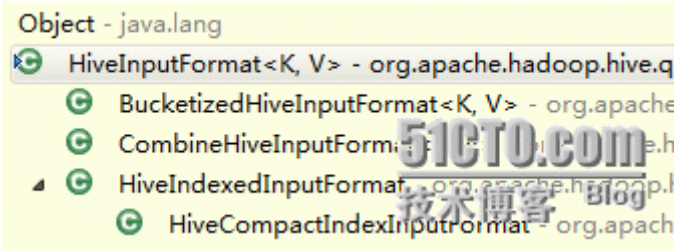
控制 Hive MAP 个数详解

> 作者：MIKE 老毕 作者：http://boylook.blog.51cto.com/7934327/1316432

Hive 的 MAP 数或者说 MAPREDUCE 的 MAP 数是由谁来决定的呢？inputsplit size,那么对于每一个 inputsplit size 是如何计算出来的，这是做 MAP 数调整的关键.

HADOOP 给出了 Inputformat 接口用于描述输入数据的格式，其中一个关键的方法就是 getSplits，对输入的数据进行分片.

Hive 对 InputFormat 进行了封装：



而具体采用的实现是由参数 hive.input.format 来决定的，主要使用 2 中类型 HiveInputFormat 和 CombineHiveInputFormat.

对于 HiveInputFormat 来说：

```
public InputSplit[] getSplits(JobConf job, int numSplits) throws IOException {
//扫描每一个分区
for (Path dir : dirs) {
   PartitionDesc part = getPartitionDescFromPath(pathToPartitionInfo, dir);
//获取分区的输入格式
   Class inputFormatClass = part.getInputFileFormatClass();
```

```
   InputFormat inputFormat = getInputFormatFromCache(inputFormatClass, job);
//按照相应格式的分片算法获取分片
//注意：这里的 Inputformat 只是 old version API：org.apache.hadoop.mapred 而不是
org.apache.hadoop.mapreduce，因此不能采用新的 API，否则在查询时会报异常：Input format
must implement InputFormat.区别就是新的 API 的计算 inputsplit size（Math.max(minSize,
Math.min(maxSize, blockSize)）和老的（Math.max(minSize, Math.min(goalSize, blockSize))）不
一样；
   InputSplit[] iss = inputFormat.getSplits(newjob, numSplits / dirs.length);
   for (InputSplit is : iss) {
//封装结果，返回
      result.add(new HiveInputSplit(is, inputFormatClass.getName()));
   }
}
return result.toArray(new HiveInputSplit[result.size()]);
}
```

对于 CombineHiveInputFormat 来说的计算就比较复杂了：

```
public InputSplit[] getSplits(JobConf job, int numSplits) throws IOException {
//加载 CombineFileInputFormatShim，这个类继承了
org.apache.hadoop.mapred.lib.CombineFileInputFormat
CombineFileInputFormatShim combine = ShimLoader.getHadoopShims()
      .getCombineFileInputFormat();
if (combine == null) {
```

```
//若为空则采用 HiveInputFormat 的方式，下同
   return super.getSplits(job, numSplits);
}
Path[] paths = combine.getInputPathsShim(job);
for (Path path : paths) {
//若是外部表，则按照 HiveInputFormat 方式分片
   if ((tableDesc != null) && tableDesc.isNonNative()) {
      return super.getSplits(job, numSplits);
   }
   Class inputFormatClass = part.getInputFileFormatClass();
   String inputFormatClassName = inputFormatClass.getName();
   InputFormat inputFormat = getInputFormatFromCache(inputFormatClass, job);
   if (this.mrwork != null && !this.mrwork.getHadoopSupportsSplittable()) {
      if (inputFormat instanceof TextInputFormat) {
        if ((new CompressionCodecFactory(job)).getCodec(path) != null)
//在未开启 hive.hadoop.supports.splittable.combineinputformat(MAPREDUCE-1597)参数情况
下，对于 TextInputFormat 并且为压缩则采用 HiveInputFormat 分片算法
                        return super.getSplits(job, numSplits);
      }
   }
//对于连接式同上
   if (inputFormat instanceof SymlinkTextInputFormat) {
      return super.getSplits(job, numSplits);
```

```
    }
    CombineFilter f = null;
    boolean done = false;
Path filterPath = path;
//由参数 hive.mapper.cannot.span.multiple.partitions 控制，默认 false;如果没 true，则对每一个
partition 创建一个 pool，以下省略为 true 的处理；对于同一个表的同一个文件格式的 split 创建一个
pool 为 combine 做准备；
    if (!mrwork.isMapperCannotSpanPartns()) {
      opList = HiveFileFormatUtils.doGetWorksFromPath(
                    pathToAliases, aliasToWork, filterPath);
      f = poolMap.get(new CombinePathInputFormat(opList, inputFormatClassName));
    }
    if (!done) {
      if (f == null) {
        f = new CombineFilter(filterPath);
        combine.createPool(job, f);
      } else {
        f.addPath(filterPath);
      }
    }
}
if (!mrwork.isMapperCannotSpanPartns()) {
```

```
//到这里才调用 combine 的分片算法，继承了

org.apache.hadoop.mapred.lib.CombineFileInputFormat extends 新版本

CombineFileInputformat

   iss = Arrays.asList(combine.getSplits(job, 1));

}

//对于 sample 查询特殊处理

if (mrwork.getNameToSplitSample() != null && !mrwork.getNameToSplitSample().isEmpty()) {

   iss = sampleSplits(iss);

}

//封装结果返回

for (InputSplitShim is : iss) {

   CombineHiveInputSplit csplit = new CombineHiveInputSplit(job, is);

   result.add(csplit);

}

return result.toArray(new CombineHiveInputSplit[result.size()]);

   }
```

具体 combine 的 getSplits 算法如下：

```
public List<InputSplit> getSplits(JobContext job)

throws IOException {

      //决定切分的几个参数

if (minSplitSizeNode != 0) {

   minSizeNode = minSplitSizeNode;
```

```
} else {
    minSizeNode = conf.getLong(SPLIT_MINSIZE_PERNODE, 0);
}
if (minSplitSizeRack != 0) {
    minSizeRack = minSplitSizeRack;
} else {
    minSizeRack = conf.getLong(SPLIT_MINSIZE_PERRACK, 0);
}
if (maxSplitSize != 0) {
    maxSize = maxSplitSize;
} else {
    maxSize= = conf.getLong("mapreduce.input.fileinputformat.split.maxsize", 0);
}
       for (MultiPathFilter onepool : pools) {
    ArrayList<Path> myPaths = new ArrayList<Path>();
    // create splits for all files in this pool.
    getMoreSplits(job, myPaths.toArray(new Path[myPaths.size()]),
                          maxSize, minSizeNode, minSizeRack, splits);
}
}
```

跳到 getMoreSplits：主要是填充如下数据结构，

```
// all blocks for all the files in input set
```

```
OneFileInfo[] files;

// mapping from a rack name to the list of blocks it has

HashMap<String, List<OneBlockInfo>> rackToBlocks = new HashMap<String,

List<OneBlockInfo>>();

// mapping from a block to the nodes on which it has replicas

HashMap<OneBlockInfo, String[]> blockToNodes = new HashMap<OneBlockInfo, String[]>();

// mapping from a node to the list of blocks that it contains

HashMap<String, List<OneBlockInfo>> nodeToBlocks = new HashMap<String,

List<OneBlockInfo>>();
```

大概流程则是（这里 blockInfo 生成略过不表，可以参考 MAPREDUCE-2046）：

1.首先处理每个 Datanode 的 blockInfo，先按照>=maxsplitsize 来切分 split，剩余的再按照 blockinfo>=minSplitSizeNode 切分，其余的等和 rack 的其余 blockinfo 进行合并

2.其次对每个 Rack 进行处理：先按照>=maxsplitsize 来切分 split，剩余的再按照 blockinfo>=minSplitSizeRack 切分，其余的等和 overflow 的其余 blockinfo 进行合并

3.对于 overflow blockInfo 直接根据 maxsplitsize 来进行切分.

其余影响 MAP 数的参数比较好理解了：

1.影响在 MAPREDUCE 后是否会启动 MAP 进行文件合并

hive.merge.mapfiles,hive.merge.mapredfiles,hive.merge.size.per.task(default=256 * 1000 * 1000),hive.merge.smallfiles.avgsize(default=16 * 1000 * 1000)

2.影响是否存在 skew 开启多 MAP:

hive.groupby.skewindata=false:

当该参数有 true 时会生成 2 个 MR：

第一个 MR 的分区键是 grouping key+distinct key，通过 hash 分配到 reduce 进行第一次聚合操作

第二个 MR 的分区键则是 grouping key 进行第二次聚合；（2 个 MR 的 sort key 都是 grouping key+distinct key）

https://issues.apache.org/jira/browse/HIVE-5118

```
hive.optimize.skewjoin=false

hive.optimize.skewjoin.compiletime=false

hive.skewjoin.key=100000

hive.skewjoin.mapjoin.map.tasks=10000

hive.skewjoin.mapjoin.min.split=33554432
```

3.mapreduce 参数，是否开启 map speculative

4.bucket table.

Shell 脚本如何监控程序占用带宽？

作者：杨云 来源：http://yangrong.blog.51cto.com/6945369/1411346

众所周知，使用 iftop 能监控所有程序占用的网络带宽，一般情况下，手动执行 iftop 就可查看。但现在需要使用脚本来监控程序占用的带宽，遇到的问题真不是一点半点，现记录如下，希望能给其它运维人带来更多的帮助。

中途所遇到的难点：

1. iftop 把结果重定向到文本中，是图形格式的。

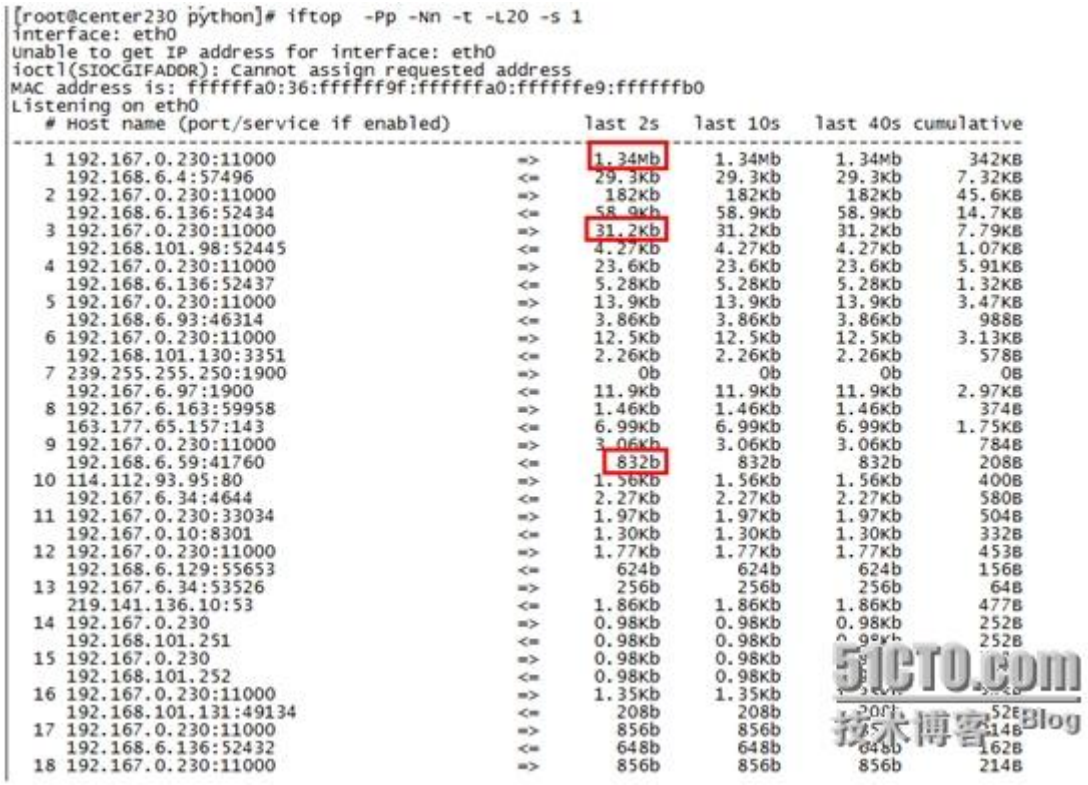
重定向到文本中的内容，全部是一行，根本无法用脚本取值。最开始我使用 python 读取这个文件，得到所有特殊符号，找到规律，然后使用 sed 替换成规范的格式。终于在自己测试机上完成，能展示出正常的格式。当放到线上机器时，特殊符号变了...又变成乱糟糟的了。网上找了很久的资料，终于找到了解决方法：iftop 1.0-pre 之后的版本都能输出文本格式，之前用的是 iftop 0.7 版本。当晚心里有种流泪的感觉，弄了一天，结果有简单现成的方法。。。

2.一个程序不仅仅只使用一个端口。

原以为程序仅仅监听一个端口进行通信，后来询问研发得知，当这个程序是服务端的时候，端口是固定的；当这个程序主动访问外面的时候，端口是随机的。所以要想监控的准确，必须找到这个程序打开的所有端口。解决方法是：用 netstat 所这个程序的所有端口找出来。

3.iftop 输出的流量单位不一样，且没有调整一致的命令。

单位不一样，里面有 Mb,Kb,b 单位，需要进行换算。我的解决方法是：把 Mb 替换成*1000,把 Kb 替换成空，把 b 直接不要过滤掉。最后用 bc 一算直接得结果。



4.程序发送占用带宽好算，接收带宽不好算。

根据第 2 步找到的几个端口，过滤出发送出去的流量一加就可以。但是接收的怎么算？见上边图中第一条流量，有"<="的则为接收流量，"<="这些行都是未知的 IP 与端口，怎么把它过滤出来得出结果？？我的解决方法是：把"=>"行和"<="放两个临时文件中，图中有"=>"的行第一列都有序号，那么全部是"<="行的都和它一一对应，如：发送"=>"中的是序号 12，13，15。那么"<="文件中的第 12，13，15 行就是对应的接收流量，是不是理解了？

5.shell 脚本代码如下

```
#!/bin/sh

#author:yangrong

#mail:10286460@qq.com

#date:2014-05-14

file_name="test.txt"
```

```
temp_file1="liuliang.txt"

temp_file2="liuliang2.txt"

iftop    -Pp -Nn -t -L 100 -s 1 >$temp_file1

pragrom_list=(VueDaemon VueCenter VueAgent VueCache VueSERVER VUEConnector

Myswitch Slirpvde)

#pragrom_list=(VueSERVER VueCenter)

>$file_name

for i in ${pragrom_list[@]}

do

    port_list=`netstat -plnt|grep $i|awk '{print $4}'|awk -F: '{print $2}'`

    port_all=""

    for port in $port_list

    do

            port_all="${port}|${port_all}"

            port_all=`echo $port_all|sed 's/\(.*\)|$/\1/g'`

    done

    if [[ $port_all == "" ]];then

            echo "${i}sendflow=0" >> $file_name

            echo "${i}receiveflow=0" >> $file_name

            continue

  fi

    send_flow=`cat $temp_file1 |grep -E "${port_all}"|grep -E 'Mb|Kb'|grep '=>'|awk '{print $4}'|\

    tr '\n' '+' |sed -e s/Mb/*1000/g |sed s/Kb//g |sed    's/\(.*\)+$/\1\n/g'|bc`
```

```
    #echo "cat liuliang.txt |grep -E "${port_all}"|grep -E 'Mb|Kb'|grep '=>'|awk '{print $4}'|\
    #tr '\n' '+' |sed -e s/Mb/*1000/g |sed s/Kb//g |sed    's/\(.*\)+$/\1\n/g'|bc"
    if [[ ${send_flow} == "" ]];then
            send_flow=0
    fi
    send_num=`cat $temp_file1 |grep -E "${port_all}"|grep "=>"|awk '{print $1}'`
    echo "" > $temp_file2
    for num in $send_num
    do
      cat $temp_file1 |grep    '<='|sed -n ${num}p|grep -E 'Mb|Kb' >>$temp_file2
    done
    receive_flow=`cat $temp_file2 |grep -E 'Mb|Kb'|awk '{print $4}'|\
    tr '\n' '+' |sed -e s/Mb/*1000/g |sed s/Kb//g |sed    's/\(.*\)+$/\1\n/g'|bc`
    if [[ $receive_flow == "" ]];then
            receive_flow=0
    fi
    echo "${i}sendflow=${send_flow}" >>$file_name
    echo "${i}receiveflow=${receive_flow}" >>$file_name
done
```

6.shell 脚本执行效果

脚本中定义的进程列表为：pragrom_list=(VueDaemonVueCenter VueAgent VueCache VueSERVER VUEConnector Myswitch Slirpvde)

执行脚本的输出单位是 Kb。

```
[root@center230 python]# rm -f test.txt
[root@center230 python]# sh flow.sh
interface: eth0
Unable to get IP address for interface: eth0
ioctl(SIOCGIFADDR): Cannot assign requested address
MAC address is: ffffffa0:36:ffffff9f:ffffffa0:ffffffe9:ffffffb0
[root@center230 python]# cat test.txt
VueDaemonsendflow=0
VueDaemonreceiveflow=0
VueCentersendflow=0
VueCenterreceiveflow=0
VueAgentsendflow=0
VueAgentreceiveflow=0
VueCachesendflow=0
VueCachereceiveflow=0
VueSERVERsendflow=1677.79
VueSERVERreceiveflow=109.86
VUEConnectorsendflow=0
VUEConnectorreceiveflow=0
Myswitchsendflow=0
Myswitchreceiveflow=0
Slirpvdesendflow=0
Slirpvdereceiveflow=0
[root@center230 python]#
```


7.附：iftop 命令用法

```
[root@center230 python]# iftop --help

iftop: unknown option --

iftop: display bandwidth usage on aninterface by host


Synopsis: iftop -h | [-npblNBP] [-iinterface] [-f filter code]
                                [-F net/mask][-G net6/mask6]


 -h                  display thismessage #帮助信息

 -n                  don't do hostname lookups   #禁用主机解析，即不会出现 IP 显示域名

 -N                   don't convertport numbers to services #以数字为示端口号，如 21 端口不会
显示成 ftp

 -p                  run inpromiscuous mode (show traffic between other
```

hosts on the samenetwork segment)

-b don't displaya bar graph of traffic #以 b 单位显示

-B Displaybandwidth in bytes #以 B 单位显示

-iinterface listen on namedinterface #指定监听的网口

-ffilter code use filter code toselect packets to count

(default: none, but onlyIP packets are counted)

-Fnet/mask show traffic flowsin/out of IPv4 network #显示指定 Ipv4 段流量

-Gnet6/mask6 show traffic flowsin/out of IPv6 network #显示指定 Ipv6 段流量

-l display andcount link-local IPv6 traffic (default: off) #显示 Ipv6 的流量

-P show ports aswell as hosts #显示端口信息

-mlimit sets the upper limit forthe bandwidth scale

-cconfig file specifies an alternativeconfiguration file

-t use textinterface without ncurses #使用文本模式输出

Sorting orders:

-o2s Sort by first column(2s traffic average) #按 2s 平均流量列排序

-o10s Sort by second column(10s traffic average) [default] #按 10s 平均流量列排序

-o40s Sort by third column(40s traffic average) #按 50s 平均流量列排序

-osource Sort by source address #按源 IP 列排序

-odestination Sort by destinationaddress #按目的 IP 列排序

The following options are only available in combination with -t

-snum print one single textoutput afer num seconds, then quit #指定刷新几次。

-Lnum number of lines to print #显示多少行数据。当程序多流量大时，则要显示行数多些才行。

iftop, version 1.0pre4 #版本信息。

文本输出方法：

iftop -Pp -Nn -t -L 100 -s 1 >temp_file

直接查看输 iftop 即可。

iftop 详细用法见网上文档。

http://www.vpser.net/manage/iftop.html

总结：

1、先尽可能的寻找已有方法。

2、基本功要杂实，对 sed,awk,grep 等命令要熟练使用。

3、思路要灵活多变，不能被一种方法束缚死。

Web 服务之 LNMMP 架构及动静分离实现

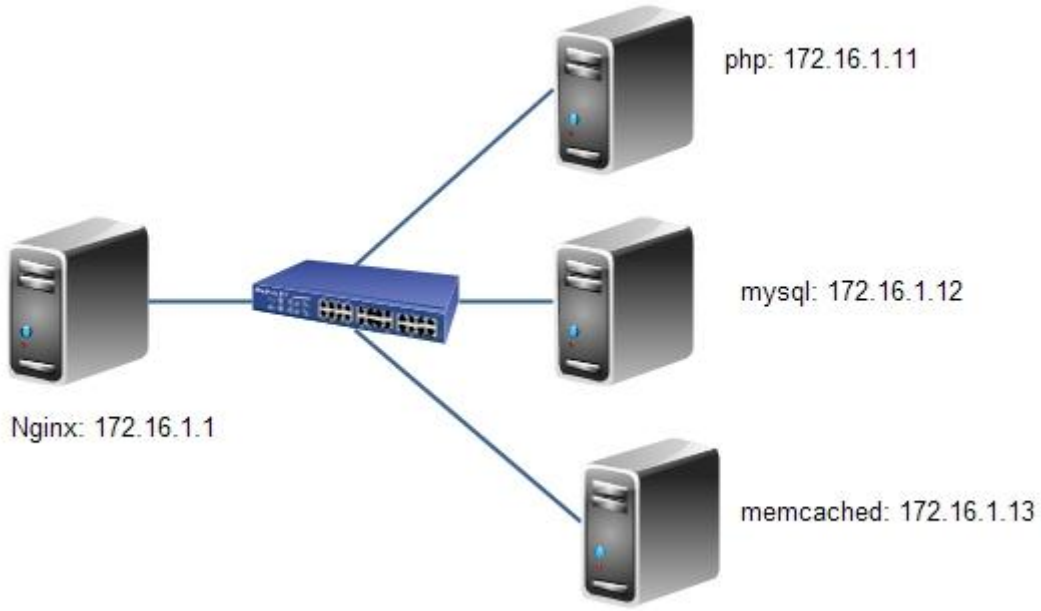
作者：hoo_5 来源：http://hoolee.blog.51cto.com/7934938/1413346

一、LNMMP

LNMMP 环境是 Linux + Nginx + Memcached + MySQL + PhP，即 LNMP + memcached。

Memcached 是一个高性能的分布式内存对象缓存系统，用于动态 Web 应用以减轻数据库负载。它通过在内存中缓存数据和对象来减少读取数据库的次数，从而提供动态、数据库驱动网站的速度。Memcached 基于一个存储键/值对的 hashmap。其守护进程（daemon ）是用 C 写的，但是客户端可以用任何语言来编写，并通过 memcached 协议与守护进程通信。

二、工程拓扑



三、安装 nginx 服务器

1）部署开发环境

yum -y install "Development tools" "Server Platform Development"

2) 解决依赖 pcre-devel openssl-devel

yum -y install pcre-devel openssl-devel

3) 设置用户

groupadd -r nginx

useradd -r -g nginx nginx

4) 编译安装 nginx-1.4.7

tar xf nginx-1.4.7.tar.gz

cd nginx-1.4.7

./configure --prefix=/usr --sbin-path=/usr/sbin/nginx \

--conf-path=/etc/nginx/nginx.conf --error-log-path=\ /var/log/nginx/error.log --http-log-

path=/var/log/nginx/access.log \

--pid-path=/var/run/nginx/nginx.pid --lock-path=/var/lock/nginx.lock \--user=nginx --

group=nginx --with-http_ssl_module \

--with-http_flv_module --with-http_stub_status_module \

--with-http_gzip_static_module --http-client-body-temp-path=\ /var/tmp/nginx/client/ --http-

proxy-temp-path=/var/tmp/nginx/proxy/ \ --http-fastcgi-temp-path=/var/tmp/nginx/fcgi/ \

--http-uwsgi-temp-path=/var/tmp/nginx/uwsgi --http-scgi-temp-path=\

/var/tmp/nginx/scgi --with-pcre

make && make install

5) 检测配置文件语法

/usr/sbin/nginx -t

6) 提供启动脚本

```
# vim   /etc/rc.d/init.d/nginx

    内容如下

# nginx - this script starts and stops the nginx daemon

#

# chkconfig:    - 85 15

# description:   Nginx is an HTTP(S) server, HTTP(S) reverse \

#                       proxy and IMAP/POP3 proxy server

# processname: nginx

# config:          /etc/nginx/nginx.conf

# config:          /etc/sysconfig/nginx

# pidfile:        /var/run/nginx.pid


# Source function library.

. /etc/rc.d/init.d/functions


# Source networking configuration.

. /etc/sysconfig/network


# Check that networking is up.

[ "$NETWORKING" = "no" ] && exit 0


nginx="/usr/sbin/nginx"

prog=$(basename $nginx)
```

```
NGINX_CONF_FILE="/etc/nginx/nginx.conf"

[ -f /etc/sysconfig/nginx ] && . /etc/sysconfig/nginx

lockfile=/var/lock/subsys/nginx

    make_dirs() {
  # make required directories
  user=`nginx -V 2>&1 | grep "configure arguments:" | sed 's/[^*]*--user=\([^ ]*\).*/\1/g' -`
  options=`$nginx -V 2>&1 | grep 'configure arguments:'`
  for opt in $options; do
  if [ `echo $opt | grep '.*-temp-path'` ]; then
      value=`echo $opt | cut -d "=" -f 2`
      if [ ! -d "$value" ]; then
          # echo "creating" $value
          mkdir -p $value && chown -R $user $value
      fi
  fi
  done
}

start() {
```

```
[ -x $nginx ] || exit 5

[ -f $NGINX_CONF_FILE ] || exit 6

make_dirs

echo -n $"Starting $prog: "

daemon $nginx -c $NGINX_CONF_FILE

retval=$?

echo

[ $retval -eq 0 ] && touch $lockfile

return $retval

}


stop() {

echo -n $"Stopping $prog: "

killproc $prog -QUIT

retval=$?

echo

[ $retval -eq 0 ] && rm -f $lockfile

return $retval

}


restart() {

configtest || return $?

stop
```

```
sleep 1

start

}


reload() {

configtest || return $?

echo -n $"Reloading $prog: "

killproc $nginx -HUP

RETVAL=$?

echo

}


force_reload() {

restart

}


configtest() {

    $nginx -t -c $NGINX_CONF_FILE

}


rh_status() {

status $prog

}
```

```
rh_status_q() {
rh_status >/dev/null 2>&1
}

 case "$1" in
start)
        rh_status_q && exit 0
        $1
        ;;
stop)
        rh_status_q || exit 0
        $1
        ;;
restart|configtest)
        $1
        ;;
reload)
        rh_status_q || exit 7
        $1
        ;;
force-reload)
        force_reload
```

```
        ;;
status)
        rh_status
        ;;
condrestart|try-restart)
        rh_status_q || exit 0
            ;;
*)
        echo $"Usage: $0 {start|stop|status|restart|condrestart|try-restart|reload|force-
reload|configtest}"
        exit 2
esac
```

7)为服务脚本赋予执行权限

```
# chmod +x /etc/rc.d/init.d/nginx
```

8) 添加到系统服务并开机启动

```
# chkconfig --add nginx
# chkconfig nginx on
# chkconfig --list nigx
```

9) 设置 nginx 配置文件的语法高亮

```
# mkdir ./vim/syntax -pv
# cd    .vim/syntax
# wget http://www.vim.org/scripts/download_script.php?src_id=19394
# cd .vim
```

```
# vim filetype.vim  内容如下

 au BufRead,BufNewFile /etc/nginx/*,/usr/local/nginx/conf/* if &ft == '' | setfiletype nginx |

endif

10 ) 启动服务

# service nginx start

# ss -tnalp | grep nginx
```

四、安装 MySQL 服务器

1.安装

```
# tar xf mysql-5.5.33-linux2.6-x86_64.tar.gz -C /usr/local

# ln -sv /usr/local/mysql-5.5.33-linux2.6-x86_64 mysql  创建软连接，易于操作
```

2.为数据库创建数据目录

```
# mkdri -pv /mydata/data
```

3.新建用户以安全方式运行进程

```
#groupadd -r mysql          //创建系统组 mysql

#useradd -r -s /sbin/nologin -g mysql mysql -M -D /mydata/data mysql

//创建系统用户 mysql

#chown -R mysql:mysql /mydata/data

//设置目录属主属组
```

4.初始化 mysql

```
# cd /usr/local/mysql

# scripts/mysql_install_db --datadir=/mydata/data --user=mysql

//初始化数据库

# chown -R root .

//设置当前目录所有文件属主为 root
```

5.提供脚本

```
#cd /usr/local/mysql

#cp support-files/mysql.server    /etc/rc.d/init.d/mysqld

//设置脚本 mysqld

#chmod +x /etc/rc.d/init.d/mysqld

//给脚本执行权限

# chkconfig --add mysqld

//添加开机启动

# chkconfig    mysqld on
```

6.提供配文件

```
#cd /usr/local/mysql

#cp support-files/my-large.cnf    /etc/my.cnf

#vim /etc/my.cnf

thread_concurrency = 2

//修改，并发线程数，bithread_concurrency 的值为 CPU 个数乘以 2

datadir = /mydata/data
```

```
#添加,mysql 数据文件的存放路径：
```

7.其他配置

```
# vim /etc/profile.d/mysqld.sh

exportPATH=/usr/local/mysql/bin:$PATH

# source /etc/profile.d/mysqld.sh

#vim /etc/man.config

MANPATH    /usr/local/mysql/man//添加此行

# ln -sv /usr/local/mysql/include    /usr/include/mysql

//输出 mysql 的头文件至系统头文件路径/usr/include

# echo '/usr/local/mysql/lib' > /etc/ld.so.conf.d/mysql.conf

//输出 mysql 的库文件给系统库

#ldconfig     //重载系统库：
```

8.启动服务

```
# service mysqld start

# ss    -tnl | grep 3306
```

9.用户初始化

```
#mysql

mysql> use mysql

mysql> selecthost,user,password from user;

mysql> DELETE FROM user WHERE user = '';       //删除空用户
```

```
mysql> DELETE FROM user WHERE user = '::1'; //删除 ipv6 用户

mysql> UPDATE user SET password = PASSWORD('Hoolee') WHERE password = '';

//为 root 用户设置密码

mysql> FLUSH PRIVILEGES;
```

五、安装 php 服务器

1.解决开发环境和依赖关系

```
# yum -y install bzip2-devel

# yum -y install libmcrypt-devel

# yum -y groupinstall "Desktop Platform Development"
```

2.安装 php

```
# tar xf php-5.4.26.tar.bz2

# cd /usr/src/php-5.4.26/

# ./configure--prefix=/usr/local/php --with-openssl

--with-mysql=mysqlnd --with-mysqli=mysqlnd --with-pdo-mysql=mysqlnd

--enable-mbstring --with-freetype-dir --with-jpeg-dir

--with-png-dir --with-zlib--with-libxml-dir=/usr --enable-xml

--enable-sockets --with-apxs2=/usr/local/apache2/bin/apxs

--with-mcrypt   --with-config-file-path=/etc --with-config-file-scan-dir=/etc/php.d

--with-bz2   --enable-maintainer-zts

# make   && make install
```

3.提供配置文件

```
# cp php.ini-production /etc/php.ini
```

4.为 php-fpm 提供脚本，并将其添加到服务列表

```
# cp sapi/fpm/init.d.php-fpm /etc/rc.d/init.d/php-fpm

# chmod +x /etc/rc.d/init.d/php-fpm

# chkconfig --add php-fpm

# chkconfig php-fpm on
```

5.为 php-fpm 提供配置文件

```
# cp /usr/local/php/etc/php-fpm.conf.default /usr/local/php/etc/php-fpm.conf
```

6.编辑 php-fpm 配置文件

```
# vim /usr/local/php/etc/php-fpm.conf

pm.max_children = 150

pm.start_servers = 8

pm.min_spare_servers = 5

pm.max_spare_servers = 10

pid = /usr/local/php/var/run/php-fpm.pid

listen = 172.16.1.11:9000
```

7.启动 php-fpm

```
# service php-fpm start

# ps -aux | grep php-fpm
```

六、安装 php 加速器 xcache

1.安装 xcache

```
# tar xf xcache-3.0.3.tar.gz

# cd xcache-3.0.3

# /usr/local/php/bin/phpize

//phpize 是用来安装 php 扩展模块的，通过 phpize 可以建立 php 的

外挂模块，若你想在原来编译好的 php 中加入 memcached 或者

ImageMagick 等扩展模块，就需要使用 phpize

# # ./configure --enable=xcache --with-php-config=/usr/local/php/bin/

php-config

# make && make install

//显示 xcache 模块路径：/usr/local/php/lib/php/extensions/no-debug-zts-20100525/
```

2.编辑配置文件，整合 php + xcache

```
# vim xcache.ini

//添加模块路径

extension = /usr/local/php/lib/php/extensions/no-debug-zts-20100525/xcache.so

# cp xcache.ini /etc/php.d/
```

3.重启 php-fpm

```
# service php-fpm restart
```

七、配置 nginx

1.编辑/etc/nginx/nginx.conf,实现动静分离

```
worker_processes  2; #worker 进程的个数
error_log  /var/log/nginx/error.log  notice;     #错误日志路径及级别
events {
worker_connections  1024;     #每个 worker 能够并发响应的最大请求数
}
http {
include          mime.types;     #支持多媒体类型
default_type  application/octet-stream;
sendfile           on;     #由内核直接转发
#keepalive_timeout  0;
keepalive_timeout  5;     #持久连接 5s
gzip  on;     #开启压缩功能
server {
    listen          80;
    server_name  www.hoo.com;
    add_header X-via $server_addr;     #让客户端能够看到代理服务器的 IP
    location / {
        root    html;
```

```
        index   index.php index.html index.htm;
    }
    location ~* \.(jpg|jpeg|png|gif|js|css)$ {   #匹配静态内容
        root    html;      #默认目录在/usr/local/nginx/html
    }
    location ~ \.php$ {   #匹配动态 php 文件
        root                    html;
        fastcgi_pass            172.16.7.11:9000;      #代理到的服务器
        fastcgi_index           index.php;
        fastcgi_param           SCRIPT_FILENAME scripts$fastcgi_script_name;
        include                 fastcgi_params;
    }
    }
}
```

2.编辑/etc/nginx/fastcgi_params

```
fastcgi_param   GATEWAY_INTERFACE   CGI/1.1;
fastcgi_param   SERVER_SOFTWARE     nginx;
fastcgi_param   QUERY_STRING        $query_string;
fastcgi_param   REQUEST_METHOD      $request_method;
fastcgi_param   CONTENT_TYPE        $content_type;
fastcgi_param   CONTENT_LENGTH      $content_length;
fastcgi_param   SCRIPT_FILENAME     $document_root$fastcgi_script_name;
```

```
fastcgi_param   SCRIPT_NAME         $fastcgi_script_name;
fastcgi_param   REQUEST_URI         $request_uri;
fastcgi_param   DOCUMENT_URI        $document_uri;
fastcgi_param   DOCUMENT_ROOT       $document_root;
fastcgi_param   SERVER_PROTOCOL     $server_protocol;
fastcgi_param   REMOTE_ADDR         $remote_addr;
fastcgi_param   REMOTE_PORT         $remote_port;
fastcgi_param   SERVER_ADDR         $server_addr;
fastcgi_param   SERVER_PORT         $server_port;
fastcgi_param   SERVER_NAME         $server_name;
```

3.重载 nginx

```
# service nginx reload
```

八、安装 Memcached 服务器

1.memcached 特性

Memcached 是一款开发工具，它既不是一个代码加速器，也不是数据库中间件。其设计哲学思想主要反映在如下方面：

（1）简单 key/value 存储：服务器不关心数据本身的意义及结构，只要是可序列化数据即可。存储项由“键、过期时间、可选的标志及数据”四个部分组成；

（2）功能的实现一半依赖于客户端，一半基于服务器端：客户负责发送存储项至服务器端、从服务端获取数据以及无法连接至服务器时采用相应的动作；服务端负责接收、存储数据，并负责数据项的超时过期；

（3）各服务器间彼此无视：不在服务器间进行数据同步；

（4）O(1)的执行效率

（5）清理超期数据：默认情况下，Memcached 是一个 LRU 缓存，同时，它按事先预订的时长清理超期数据；但事实上，memcached 不会删除任何已缓存数据，只是在其过期之后不再为客户所见；而且，memcached 也不会真正按期限清理缓存，而仅是当 get 命令到达时检查其时长；

2.安装 memcached

a).部署开发环境，解决依赖关系

```
# yum groupinstall "Development Tools" "Server Platform Deveopment" -y
# yum install -y libevent-devel
```

b).编译安装 memcached

```
# tar xf memcached-1.4.15.tar.gz
# cd memcached-1.4.15
# ./configure --prefix=/usr/local/memcached --with-libevent=/usr/local/libevent
# make && make install
```

c).为 memcached 提供启动脚本

```
#!/bin/bash
#
# Init file for memcached
#
# chkconfig: - 86 14
# description: Distributed memory caching daemon
#
# processname: memcached
# config: /etc/sysconfig/memcached
. /etc/rc.d/init.d/functions
## Default variables
PORT="11211"
USER="nobody"
MAXCONN="1024"
CACHESIZE="64"
OPTIONS=""
RETVAL=0
prog="/usr/local/memcached/bin/memcached"
desc="Distributed memory caching"
lockfile="/var/lock/subsys/memcached"
start() {
        echo -n $"Starting $desc (memcached): "
        daemon $prog -d -p $PORT -u $USER -c $MAXCONN -m $CACHESIZE -o "$OPTIONS"
```

```
        RETVAL=$?
        [ $RETVAL -eq 0 ] && success && touch $lockfile || failure
        echo
        return $RETVAL
}
stop() {
        echo -n $"Shutting down $desc (memcached): "
        killproc $prog
        RETVAL=$?
        [ $RETVAL -eq 0 ] && success && rm -f $lockfile || failure
        echo
        return $RETVAL
}
restart() {
        stop
        start
}
reload() {
        echo -n $"Reloading $desc ($prog): "
        killproc $prog -HUP
        RETVAL=$?
        [ $RETVAL -eq 0 ] && success || failure
        echo
```

```
        return $RETVAL
}
case "$1" in
    start)
        start
        ;;
    stop)
        stop
        ;;
    restart)
        restart
        ;;
    condrestart)
        [ -e $lockfile ] && restart
        RETVAL=$?
        ;;
    reload)
        reload
        ;;
    status)
        status $prog
        RETVAL=$?
        ;;
```

```
    *)
      echo $"Usage: $0 {start|stop|restart|condrestart|status}"
      RETVAL=1
esac
exit $RETVAL
```

d).添加 memcached 至服务列表

```
# chmod +x /etc/init.d/memcached
# chkconfig --add memcached
# service memcached start
```

e).检查是否运行

```
# ss -tnlp | grep 11211
```

九、安装 php 的 memcached 扩展

1.安装 memcached 扩展

```
# tar xf memcache-2.2.7.tgz
# cd memcache-2.2.7
# /usr/local/php/bin/phpize
# ./configure --with-php-config=/usr/local/php/bin/php-config --enable-memcache
# make && make install
```

2.编辑配置文件，整合 php + memcached

```
# vim xcache.ini

//添加模块路径

extension = /usr/local/php/lib/php/extensions/no-debug-zts-20100525/memcache.so
```

3.重启 php-fpm

```
# service php-fpm restart
```

4.对 memcached 功能进行测试，在网站目录中建立测试页面 test.php

```
//php 服务器

# vim test.php

<?php

$mem = new Memcache;

$mem->connect("172.16.1.13", 11211)  or die("Could not connect");

$version = $mem->getVersion();

echo "Server's version: ".$version."<br/>\n";

$mem->set('hellokey', 'Hello World', 0, 600) or die("Failed to save data at the memcached

server");

echo "Store data in the cache (data will expire in 600 seconds)<br/>\n";

$get_result = $mem->get('hellokey');

echo "$get_result is from memcached server.";

?>

~
```

测试访问即可。

5.安装 wordpress 测试访问即可

十、安装 memadmin-master 查看 memcached 状态信息

1.解压即可使用

```
//解压至 php 服务器/usr/local/nginx/html/

# unzip memadmin-master.zip
```

2.测试访问

www.hoo.com/memadmin-master

zabbix 企业应用之报表功能

作者：dl528888 来源：http://dl528888.blog.51cto.com/2382721/1410984

对于运维来说,监控是一个重要的工作，如果做好了监控可以解决以下问题：

1、做了硬件监控，如果服务器出现硬件问题可以提前知晓，提前安排好解决方案，避免突然出现问题造成损失；

2、做了系统与服务的监控，如果系统资源与服务出现问题，可以及时知晓并解决，同时可以根据周期内监控数据，做好调优；

如果仅完成以上事情的话，只是对运维本身工作有所帮忙，如何对其他部门做支持，以及让公司领导看出运维团队的重要性，就需要多下一份功夫，毕竟如果出现问题，就是运维工作不到位，如果不出问题，是运维应该做的。

为了提供运维团队对其他部门的支持，以及为运维争取话语权，我除了对以上 2 个工作更好、快速的完成外，还对于监控数据充分利用起来，通过监控数据实现报表功能，实现以下工作：

1、运维无法直接创造利益，就只能开源节流，节省服务器数量，合并压力小的业务，以便节省服务器数量与成本；

2、通过监控数据，计算出各项目使用的资源量与成本，方便各项目负责人的知晓业务使用情况与成本；

3、方便财务统计，并知晓各项目使用机房流量带宽百分百。

所以我使用 shell+mysql，写了个统计报表功能，能定时的统计每个月（我默认是每月，可以自定义时间）以下信息的信息（我统计是平均值，是概数，供参考）：

1、主机资源使用

功能：包括查询时间、主机所属组、主机 ip、cpu 逻辑核数、cpu 平均空闲值、cpu 平均最小值、可用平均内存、可用最小内存、总内存、cpu 最小 wio、cpu 最大 wio、进入最大流量、出去最大流量、进入平均流量、出去平均流量、进入最小流量、出去最小流量；

作用：使用这个表可以帮忙我们对浪费服务器资源的项目适当减少服务器数量，以便节省资源与成本。

日期	所属组	主机ip	cpu逻辑核数(单位:个)	cpu平均空闲值(单位:%)	cpu最小空闲值(单位:%)	可用平均内存(单位:GB)	可用最小内存(单位:GB)	总内存值(单位:GB)	cpu最小wio(单位:%)	cpu最大wio(单位:%)	cpu5分钟最大负载	进入最大流量(单位:kbps)	出去最大流量(单位:kbps)	进入平均流量(单位:kbps)	出去平均流量(单位:kbps)	进入最小流量(单位:kbps)	出去最小流量(单位:kbps)
2014年4月	鲁谷机房--自研项目--xxx项目	10.10.13.76	4	99.8	98.2	2.4	2.4	4	0	0.6	0	766	105	6	3	3	2
2014年4月	鲁谷机房--自研项目--xxx项目	10.10.11.98	16	96.4	83.6	5.8	5.2	16	0	2.5	6	7	2	7	2	6	2
2014年4月	鲁谷机房--自研项目--xxx项目	10.10.11.95	16	99.6	97.1	10.6	10.1	16	0	0.2	0	517	317	30	7	[illegible]	6
2014年4月	鲁谷机房--自研项目--xxx项目	10.10.11.89	16	97.2	73.8	6.8	0.5	16	0.2	3.1	7	470	8884	89	2091	[illegible]	177
2014年4月	鲁谷机房--自研项目--xxx项目	10.10.11.88	16	90.3	84.1	11.5	11.2	31	0.7	0.9	3	764	7776	385	2177	[illegible]	946
2014年4月	鲁谷机房--自研项目--xxx项目	10.10.11.87	16	99	94.7	11.3	10.5	16	0.1	4.3	2	8908	641	2097	111	180	14

2、各项目网络流量

功能：包括查询时间、主机所属组、进入最大流量、出去最大流量、进入平均流量、出去平均流量、进入最小流量、出去最小流量；

作用：方便各项目查看自己使用网络流量与计算成本。

如下图

日期	所属组	进入最大流量(单位:kbps)	出去最大流量(单位:kbps)	进入平均流量(单位:kbps)	出去平均流量(单位:kbps)	进入最小流量(单位:kbps)	出去最小流量(单位:kbps)
2014年4月	鲁谷机房--自研项目--xxx项目	2946	26955	204	1028	3	1
2014年4月	Zabbix_servers	21872	6515	13257	4659	5818	3398

3、机房网络流量

功能：包括查询时间、机房、进入最大流量、出去最大流量、进入平均流量、出去平均流量、进入最小流量、出去最小流量；

作用：方便运维了解机房网络使用量与每月机房带宽成本计算（一般机房计算机房带宽成本都使用 cacti）。

如下图

日期	机房	进入最大流量(单位:Mbps)	出去最大流量(单位:Mbps)	进入平均流量(单位:Mbps)	出去平均流量(单位:Mbps)	进入最小流量(单位:Mbps)	出去最小流量(单位:Mbps)
2014年4月	世纪互联	106	162	61	73	13	14
2014年4月	鲁谷机房	101	146	55	81	12	19

4、各项目占机房总流量百分比

功能：包括查询时间、所属组、进入最大流量、出去最大流量、进入平均流量、出去平均流量、进入最小流量、出去最小流量；

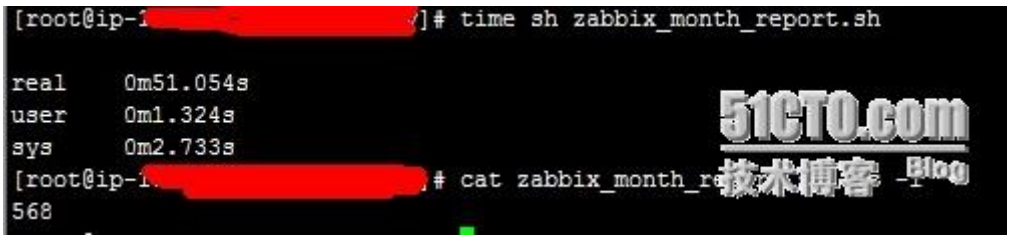
作用：能及时满足财务的智能带宽分配需求。

如下图

日期	所属组	进入最大流量(单位:%)	出去最大流量(单位:%)	进入平均流量(单位:%)	出去平均流量(单位:%)	进入最小流量(单位:%)	出去最小流量(单位:%)
2014年4月	世纪互联机房--自研项目--xxx项目	10.59	6.91	0.28	0.2	0.03	0
2014年4月	鲁谷机房--自研项目--xxx项目	2.92	18.46	0.37	1.27	0.03	0.01

下面是介绍如何实现：

1、脚本运行时间



可以看到 51 秒后就能完成。

完成后会在/tmp/zabbix_log 目录里有 4 个文件生成

```
[root@ip-10-10-13-8 zabbix_log]# pwd

/tmp/zabbix_log

[root@ip-10-10-13-8 zabbix_log]# ll
```

```
总用量 100
-rw-r--r-- 1 root root  5484 5 月  14 11:19 zabbix_group_network_traffic.txt
-rw-r--r-- 1 root root 78282 5 月  14 11:19 zabbix_host_search.txt
-rw-r--r-- 1 root root  5477 5 月  14 11:19 zabbix_network_percent.txt
-rw-r--r-- 1 root root   296 5 月  14 11:19 zabbix_room_network.txt
```

下面分别介绍一下这 4 个文件

zabbix_group_network_traffic.txt 对应“各项目网络流量”

zabbix_host_search.txt 对应“主机资源使用”

zabbix_network_percent.txt 对应“各项目占机房总流量百分比”

zabbix_room_network.txt 对应“机房网络流量”

由于运行脚本后会生成 txt 文件，非技术人员还是喜欢看 excel，所以下一步介绍如何把 txt 转为 excel

2、txt 转为 excel

请参看“http://jingyan.baidu.com/article/359911f5108f3757fe0306fb.html”，我就不介绍了，很简单。

3、脚本内容

由于脚本内容过多，我就简单介绍前几行

```
#!/bin/bash
. /etc/profile
```

```
logdir='/tmp/zabbix_log'

mysql_host='10.10.11.12'

mysql_user='zabbix'

mysql_passwd='zabbix'

mysql_database='zabbix'

year=`date +%Y`

month=`date +%m`

next_month=`echo $month+1|bc`

if [ ! -d $logdir ];then

mkdir $logdir

fi
```

默认会新建立个/tmp/zabbix_log 目录来存放 txt 文件，然后定义好了 mysql 信息，同时搜索的日期是从本月的 1 日 0 点到下月 1 日的 0 点（比如现在是 5 月，那么搜索日期是从 2014-05-01 00:00:00 到 2014-06-01 00:00:00）.

4、搜索 cpu 资源 sql

```
#select cpu avg idle

mysql -h $mysql_host   -u $mysql_user -p$mysql_passwd

$mysql_database >$logdir/info_mysql_cpu_avg_idle.txt<<EOF

set names utf8;
```

```
select from_unixtime(hi.clock,'%Y-%m') as Date,g.name as Group_Name,h.host as Host,
round(avg(hi.value_avg),1) as Cpu_Avg_Idle  from hosts_groups hg join groups g on g.groupid
= hg.groupid jo
in items i on hg.hostid = i.hostid join hosts h on h.hostid=i.hostid join trends hi on  i.itemid =
hi.itemid  where  i.key_='system.cpu.util[,idle]' and  hi.clock >= UNIX_TIMESTAMP('${year}-$
{month}-01 00:00:00') and  hi.clock < UNIX_TIMESTAMP('${year}-0${next_month}-01 00:00:00')
group by h.host;
EOF
```

5、搜索 cpu 等待 sql

```
#select cpu avg idle
mysql -h $mysql_host  -u $mysql_user -p$mysql_passwd
$mysql_database >$logdir/info_mysql_cpu_avg_idle.txt<<EOF
set names utf8;
select from_unixtime(hi.clock,'%Y-%m') as Date,g.name as Group_Name,h.host as Host,
round(avg(hi.value_avg),1) as Cpu_Avg_Idle  from hosts_groups hg join groups g on g.groupid
= hg.groupid jo
in items i on hg.hostid = i.hostid join hosts h on h.hostid=i.hostid join trends hi on  i.itemid =
hi.itemid  where  i.key_='system.cpu.util[,idle]' and  hi.clock >= UNIX_TIMESTAMP('${year}-$
{month}-01 00:00:00') and  hi.clock < UNIX_TIMESTAMP('${year}-0${next_month}-01 00:00:00')
group by h.host;
EOF
```

6、搜索 5 分钟最大负载 sql

```
#select cpu max load 5 minute
mysql -h $mysql_host   -u $mysql_user -p$mysql_passwd
$mysql_database >$logdir/info_mysql_cpu_max_load5.txt<<EOF
set names utf8;
select from_unixtime(hi.clock,'%Y-%m') as Date,g.name as Group_Name,h.host as Host,
round(max(hi.value_max),0) as Cpu_Max_Iowait   from hosts_groups hg join groups g on
g.groupid = hg.groupid
join items i on hg.hostid = i.hostid join hosts h on h.hostid=i.hostid join trends hi on   i.itemid
= hi.itemid   where   i.key_='system.cpu.load[all,avg5]' and   hi.clock >=
UNIX_TIMESTAMP('${ye
ar}-${month}-01 00:00:00') and   hi.clock < UNIX_TIMESTAMP('${year}-0${next_month}-01
00:00:00') group by h.host;
EOF
```

7、搜索平均内存 sql

```
#select memory avg avaiable
mysql -h $mysql_host   -u $mysql_user -p$mysql_passwd
$mysql_database >$logdir/info_mysql_memory_avg_avaiable.txt<<EOF
set names utf8;
```

```
select from_unixtime(hi.clock,'%Y-%m') as Date,g.name as Group_Name,h.host as
Host,round(avg(hi.value_avg)/1024/1024/1024,1) as Memory_Avaiable  from hosts_groups hg
join groups g on g.groupi
d = hg.groupid join items i on hg.hostid = i.hostid join hosts h on h.hostid=i.hostid join
trends_uint hi on  i.itemid = hi.itemid  where  i.key_='vm.memory.size[available]'  and
hi.clock >=
 UNIX_TIMESTAMP('${year}-${month}-01 00:00:00') and  hi.clock < UNIX_TIMESTAMP('${year}-
0${next_month}-01 00:00:00') group by h.host;
EOF
```

8、搜索总共内存值 sql

```
#select memory total
mysql -h $mysql_host  -u $mysql_user -p$mysql_passwd
$mysql_database >$logdir/info_mysql_memory_total.txt<<EOF
set names utf8;
select from_unixtime(hi.clock,'%Y-%m') as Date,g.name as Group_Name,h.host as
Host,round(avg(hi.value_avg)/1024/1024/1024,0) as Memory_Total  from hosts_groups hg join
groups g on g.groupid =
 hg.groupid join items i on hg.hostid = i.hostid join hosts h on h.hostid=i.hostid join
trends_uint hi on  i.itemid = hi.itemid  where  i.key_='vm.memory.size[total]' and
hi.clock >= UNIX_TI
MESTAMP('${year}-${month}-01 00:00:00') and  hi.clock < UNIX_TIMESTAMP('${year}-
0${next_month}-01 00:00:00') group by h.host;
```

```
EOF
```

9、搜索平均 em2 网卡进入与出去流量

```
#select network em2 avg in
mysql -h $mysql_host   -u $mysql_user -p$mysql_passwd
$mysql_database >$logdir/info_mysql_network_em2_avg_in.txt<<EOF
set names utf8;
 select from_unixtime(hi.clock,'%Y-%m') as Date,g.name as Group_Name,h.host as
Host,round(avg(hi.value_avg)/1000,0) as Network_Em2_Avg_In   from hosts_groups hg join
groups g on g.groupid = hg
.groupid join items i on hg.hostid = i.hostid join hosts h on h.hostid=i.hostid join trends_uint hi
on   i.itemid = hi.itemid   where   i.key_='net.if.in[em2]' and   hi.clock >= UNIX_TIMESTAMP('$
{year}-${month}-01 00:00:00') and   hi.clock < UNIX_TIMESTAMP('${year}-0${next_month}-01
00:00:00') group by h.host;
EOF
sed -i '/Date*/d' $logdir/info_mysql_network_em2_avg_in.txt
#select network em2 avg out
mysql -h $mysql_host   -u $mysql_user -p$mysql_passwd
$mysql_database >$logdir/info_mysql_network_em2_avg_out.txt<<EOF
set names utf8;
 select from_unixtime(hi.clock,'%Y-%m') as Date,g.name as Group_Name,h.host as
Host,round(avg(hi.value_avg)/1000,0) as Network_Em2_Avg_Out   from hosts_groups hg join
groups g on g.groupid = h
```

```
g.groupid join items i on hg.hostid = i.hostid join hosts h on h.hostid=i.hostid join trends_uint
hi on  i.itemid = hi.itemid  where  i.key_='net.if.out[em2]' and  hi.clock >=
UNIX_TIMESTAMP(
'${year}-${month}-01 00:00:00') and  hi.clock < UNIX_TIMESTAMP('${year}-0${next_month}-01
00:00:00') group by h.host;
EOF
sed -i '/Date*/d' $logdir/info_mysql_network_em2_avg_in.txt
paste $logdir/info_mysql_network_em2_avg_in.txt $logdir/info_mysql_network_em2_avg_out.txt
|awk '{print $1"\t"$2"\t"$3"\t"$4"\t"$NF}' >$logdir/info_mysql_network_em2_avg.txt
```

由于我公司服务器系统比较繁杂，rhel 或者 centos 5、6，ubuntu 12.04 与 12.04.4，windows 2003/2008/2012，这样导致很多监控项没办法全部查看，所以数据不是非常的精确。

关于外网流量，由于网卡名也不一样，有的网卡是 em、有的是 eth、有的是 Broadcom NetXtreme Gigabit Ethernet #2 等，并且我这里如果网卡名是 em 的话，em1 是内网，em2 是外网；网卡名是 eth 的话，eth0 是内网，eth1 是外网。

所以如果各位想使用我脚本的话，肯定得自己根据自己需求来修改，我分享脚本主要是让大家看看各个监控项的 sql，具体如何写就看各位了。

如何在一台 ESXi 主机上搭建一整套 VSAN 集群的环境

作者：delxu 来源：http://delxu.blog.51cto.com/975660/1412092

从上周起，我开始翻译一本新书。IT 类中文书籍的翻译往往有一个术语的问题，如何选择最准确的中文术语，让读者清楚明白而且在实际操作和配置的时候不至于误解，是一件不那么容易的事情。一个简单的例子就是 cluster，中文可以译作“集群”或者“群集”。这两者本身都广为使用，而且是完全相同的意思。我查了一下中文亚马逊书店，136 本 IT 类中文图书用了“集群”，29 本用了“群集”。

看上去“集群”更为普及一些，而且我本人也习惯用“集群”这个术语。那么在本书中到底应该翻译成集群还是“群集”呢？我决定用“群集”。为啥呢？因为 vSphere 中文版客户端和中文版 vSphere Web 客户端都使用“群集”作为术语。为了方便读者阅读本书的时候，能够和中文版系统和中文版帮助文件对照起来，所有能够在中文版中找到的术语，我都将用中文版本里面的术语来进行翻译。

这本新书是关于 VMware VSAN 的，为了翻译的 VSAN 术语更加准确，我需要搭建一个 VSAN 集群。可是 VSAN 集群的构建有一些必要的前提条件：

至少 3 台以上的 vSphere ESXi 5.5U1 主机

每台主机需要至少一块 SSD 和一块磁盘（至少有 3 台主机有这样的配置）提供给 VSAN 数据存储用

每台主机至少要有千兆网卡，推荐万兆。如果是千兆，建议有一个千兆端口专门提供给 VSAN 专用。

每台主机最少 6GB 内存

支持虚拟化技术的 Intel 或 AMD 处理器，至少是 Intel XEON corei7 级别或更高。

注：关于 VSAN 的安装和前提条件，可以参考一下文章和资料：

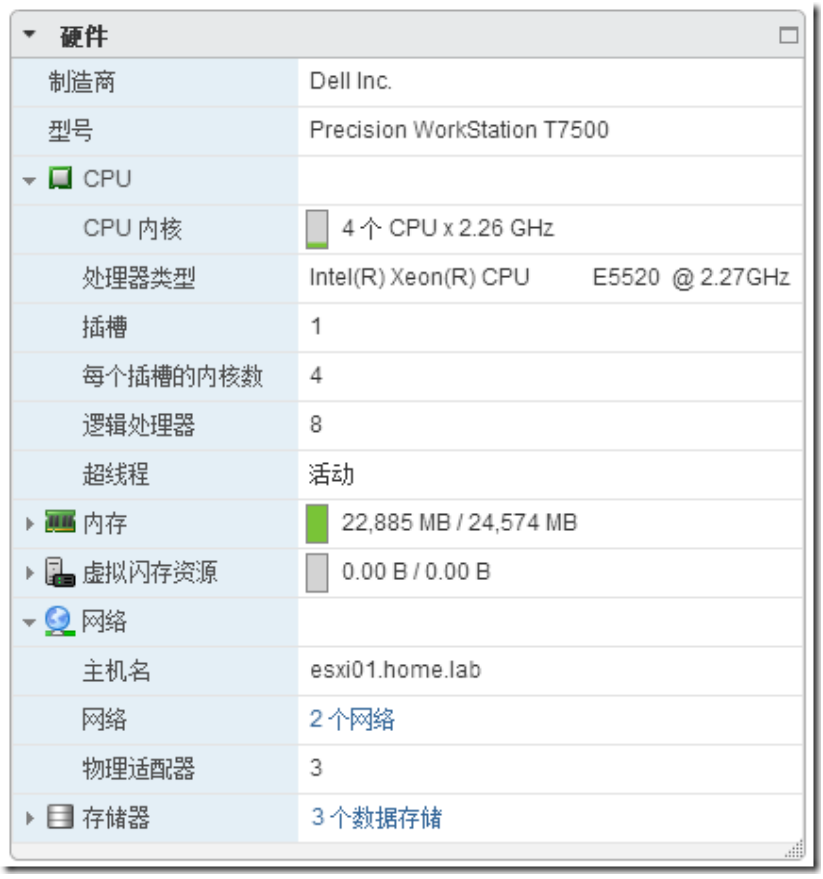
http://vsdsrevolution.blog.51cto.com/8674155/1381076

http://vsdsrevolution.blog.51cto.com/8674155/1386083

http://www.vmware.com/files/cn/pdf/products/vsan/VMware_Virtual_SAN_Whats_New.pdf

看了一下，3 台主机就有点困难，每台 2 块千兆网卡、6GB 内存，克服一下还能找到，大容量磁盘也还算容易，要搞 3 块 SSD 就难度略大。终于咬了咬牙，自己掏钱在亚马逊下单买了 2 块 SSD，加上自己家里原来有的 1 块，凑够了 3 块。不过周二拜访了 VMware 公司，VMware 中国研发中心的林博士却给了我一个建议，可以在虚拟机上模拟嘛。至于 SSD 问题，书里面第 3 章讲到某些 SSD 因为 RAID-0 控制器无法识别成 SSD 的时候，可以用命令行来强制指定为 SSD，那么在虚拟机的情况下，也可以用同样的方法糊弄一下。我恍然大悟，于是回家在自己家里的 ESXi 主机上开搞。（那 2 块 SSD 的钱啊已经花出去了！哭！）

我家里只有一台 DELL Precision T7500 的工作站用作实验用的 ESXi 主机，这台机器是在淘宝上花 4000 元淘来的 DELL 的库存货，虽然是 3 年前的硬件，但是配上 XEON 处理器和 24GB 服务器专用内存，玩个服务器虚拟化还是不错滴。



废话一大篓，现在切入正题。

接下去，我就来介绍下如何在一台 ESXi 主机上配置 3 个 ESXi 并组建一个 VSAN 集群。昨天，我介绍了如何在一台 ESXi 主机上安装 ESXi 的客户机（当然这些 ESXi 本身也是主机哦，还可以在其上部署虚拟机，虽然性能会比较差）。因此，首先就是根据上面所说的硬件条件创建 3 个虚拟机用来安装 ESXi5.5u1。我的配置是每一台主机都包括：

4 个 CPU（最少 2 个）；

8GB 内存;

3 个硬盘，一个 4GB（用来装系统）、一个 40GB（模拟成 SSD）、一个 400GB（提供给 vsan 存放数据）；

2 个网络适配器，一个在子网 192.168.10.x 用于管理和虚拟机网络，一个在子网 192.168.20.x，用于 VSAN VMkernel

虚拟机版本 10

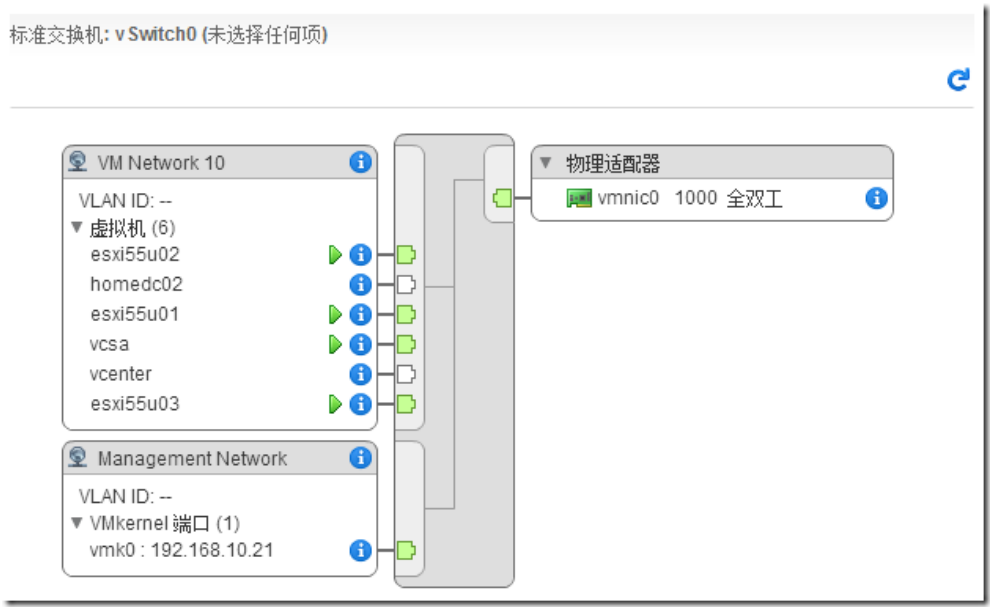
注意，为了让 ESXi 客户机有 2 个网络适配器，在 ESXi 主机（本例中起名为 esxi01）上的网络配置至少要配置 2 个不同的端口组，我将这 2 个端口组分别配置在了 2 个不同的 vSwitch 上：

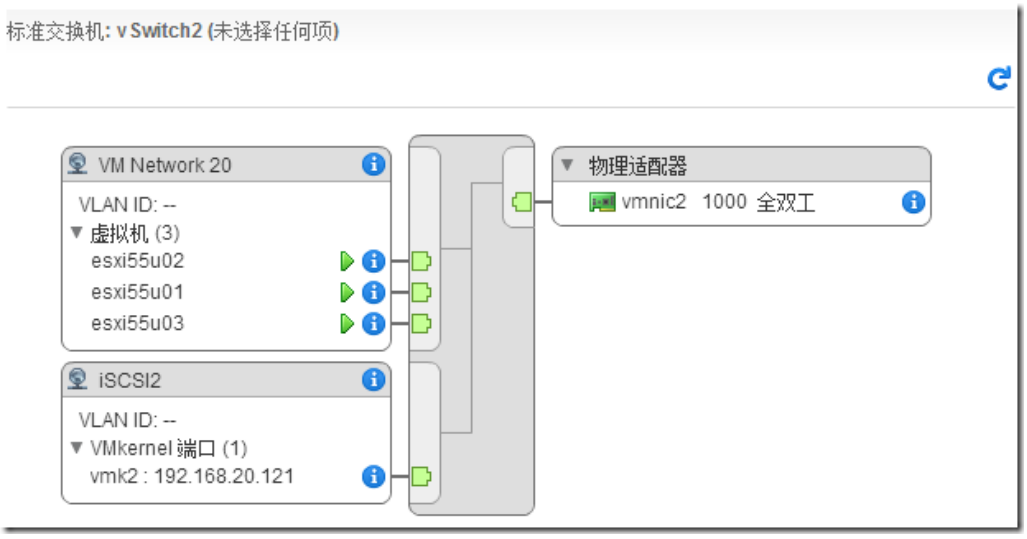
vSwitch0，默认 vSwitch，配置有管理网络(VMKernel)和 VM Network 10 端口组

vSwitch2，新增的 vSwitch，配置有 VM Network 20 端口组

此外，我还因为有 iSCSI 存储，因此配置了 2 个 iSCSI 的 VMKernel 分别在 vSwitch1 和 vSwitch2 上。

vSwitch0 和 vSwitch2 的配置截图如下：





这里有一点要说明的是，如果仅为了 vsan 网络的通信，vSwitch2 可以没有上联的物理适配器，我的截图里面配置了上联的物理适配器是因为我还有一个 iSCSI2 的 VMkernel 需要使用。

安装 3 台虚拟机的过程就不赘述了，只要你按照我昨天的文章来操作，就可以顺利完成。安装完之后，照例是配置静态 IP 地址、FQDN 主机名、禁用 IPv6、DNS 并开启 SSH 和 ESXi SHELL 以备之后在控制台输入命令行使用。

需要强调一次的是，在你的实验网络中，需要配置好一台域控制器，它同时也是 DNS 服务器，并事先在 DNS 服务器里面添加好静态的 ESXi 主机的 DNS 项。在我的实验环境中，它们是：

esxi55u01.home.lab – 192.168.10.31

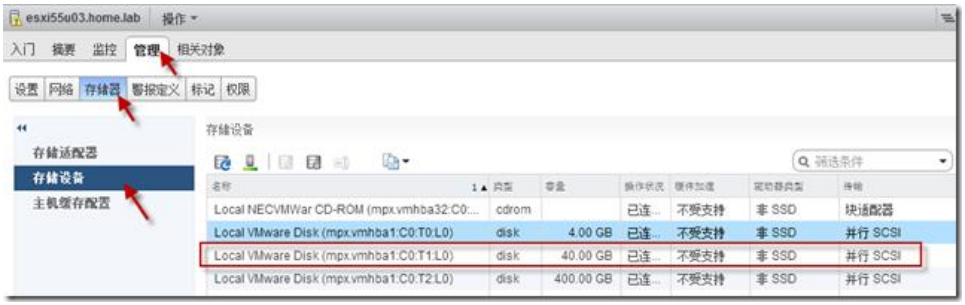
esxi55u02.home.lab – 192.168.10.32

esxi55u03.home.lab – 192.168.10.33

请在黄色 DCUI 界面（安装完 ESXi 主机的初次配置界面）里面测试一下网络，如果主机域名也能顺利解析，那就说明网络配置都完备了。DNS 对于正确配置 VMware 集群来说非常重要。

接下去就是用 vSphere Web 客户端再次连接到 vCenter（我的是 vCSA），把这几台新安装的 ESXi 主机添加进来，添加的时候要用 FQDN，不能是 IP 地址。

现在让我们来看一看这些 ESXi 主机的磁盘情况（左边窗格点选主机，在右边窗格分别选择管理，存储器和存储设备，如图所示），可以看见置备的 3 个磁盘都是非 SSD。下面要克服的问题是怎样欺骗 ESXi，让它以为其中一块 40GB 的磁盘是 SSD，这样才能满足 VSAN 配置的必要前提条件。



让我们进入到这台 vSphere ESXi 主机的管理控制台界面，在命令行里面输入下面的 2 条命令，就可以完成：

```
# esxcli storage nmp satp rule add --satp VMW_SATP_LOCAL --device mpx.vmhba1:C0:T1:L0 --option=enable_ssd
# esxcli storage core claiming reclaim -d mpx.vmhba1:C0:T1:L0
```

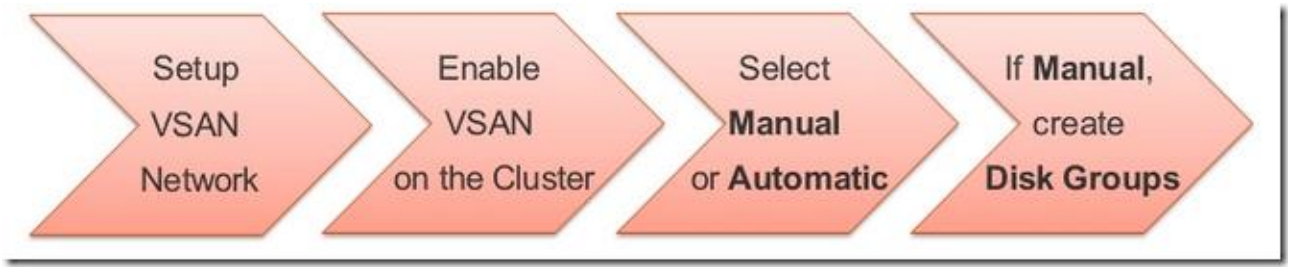
注意，这里的设备 ID 要填写你所想要变成 SSD 的那个磁盘，设备 ID 就是长的像 mpx.vmhba1.C0:T1:L0 的那个。

输入命令后，如果没有填错，是不返回结果的。回到 vSphere Web 客户端，刷新一下，你就会发现那个 40GB 的磁盘的类型变成 SSD 了。

关于 VSAN 的配置，LeoXiao 同学写的很不错，就不多罗嗦了。你可以参考他的文章。

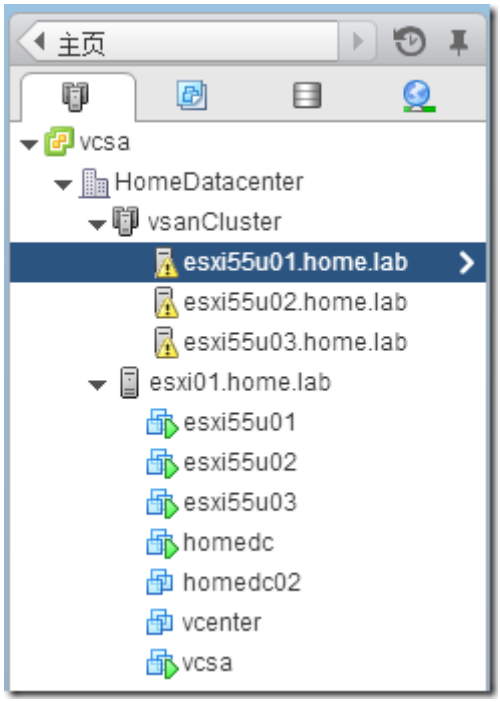
http://sanshileilei.blog.51cto.com/3105269/1375551

我借个图说明下顺序：



多说一句，为了测试 NIOC 功能，而这个功能只有在分布式交换机上才有，所以，建议 VSAN 集群配置的时候先配置好分布式交换机，并且把 VSAN 专用的 VMkernel 建在分布式交换机上。

最后，给大家看一下，要搭建一个 VSAN 集群的测试环境，在一台主机上至少要配置并开启 5 台虚拟机——包括 1 台域控制器，1 台 vCenter 和 3 台 ESXi 主机（每台都配了 8GB 内存哦）。



虽然还是有一些网络传输上不稳定的问题，不过 vsan 数据存储好歹是建起来了。

最后的感慨是 24GB 内存的主机还是缺内存，555，性能还是好差啊 ~

微软私有云分享（R2）-SCVMM 报错干货一小波

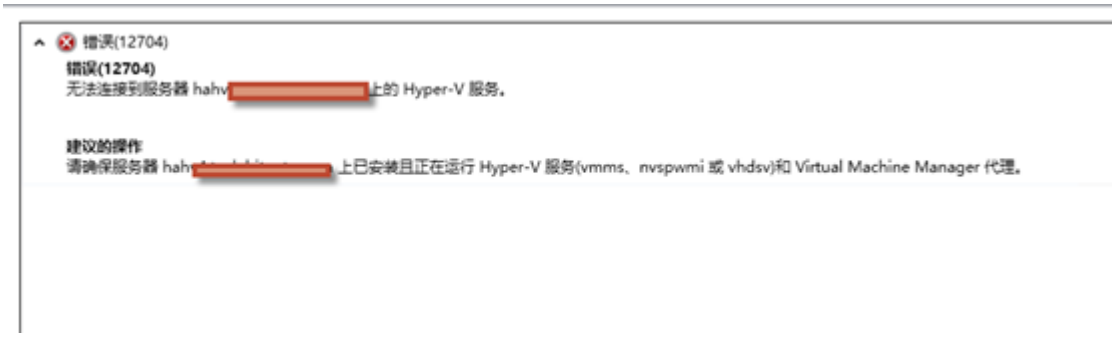
作者：九叔 来源：http://jiushu.blog.51cto.com/972756/1411992

对于运维工作而言，稳定运行当然没啥说的，让人头痛的都是出现了一大波错误却无法解决。

今天为大家分享一下我在 SCVMM2012（SP1 和 R2） 中遇到的一大波错误。

=======都是干货==========

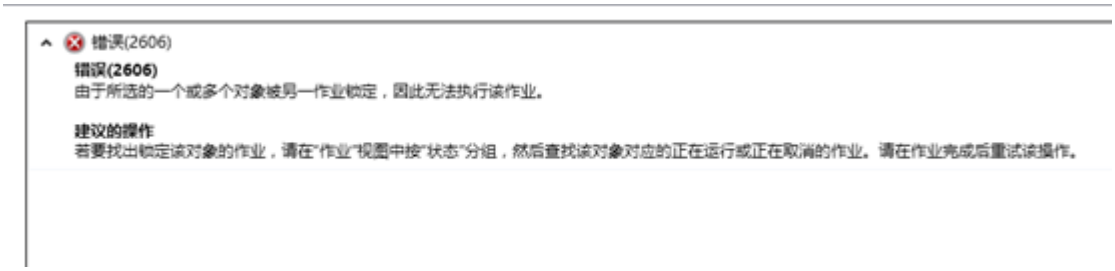
12704，无法连接到 Hyper-V 服务，一般来说这时候是网络有问题造成 SCVMM2012 R2 与 Hyper-V 主机主机失去联系。这时候要先排查网络，看用 Hyper-V 能否连接，连接不了就重启机器吧。



25322，群集悲剧，新增加的群集节点配置和其他节点不一样，这个就是看提示，慢慢让全新节点和老节点配置一致。

2927，这个问题出现的很诡异，一般来说都是 SCVMM2012 服务器挂了，多数情况下重启 SCVMM2012 或者 SCVMM2012 R2 所在的虚拟机就可以了，如果还不好使，需要同时重启 SCVMM2012 R2 所用到的 sql 服务器。

2606，业务被锁定，这个原因很多，主要是看具体是什么业务正在运行，导致其他关联的业务无法重启。懒人做法就是重启 SCVMM2012 R2 服务器。



12703，添加 xenserver 时才会遇到的问题，这个时候需要导入一个注册表文件。

Reg 文件代码具体如下，导入即可，具体为啥你也别问我

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL]

"EventLogging"=dword:00000001

"SendExtraRecord"=dword:00000002

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers]

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\CipherSuites]

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Hashes]
```

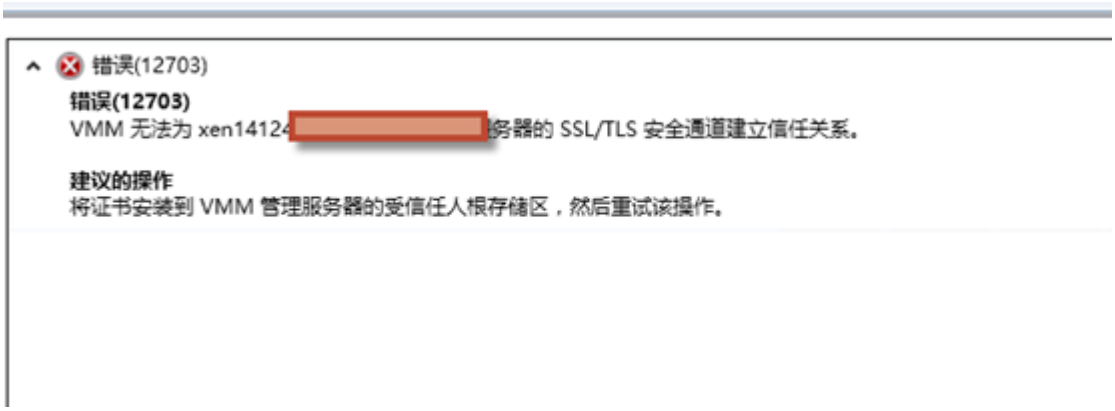
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\KeyExchangeAlgorithms]

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols]

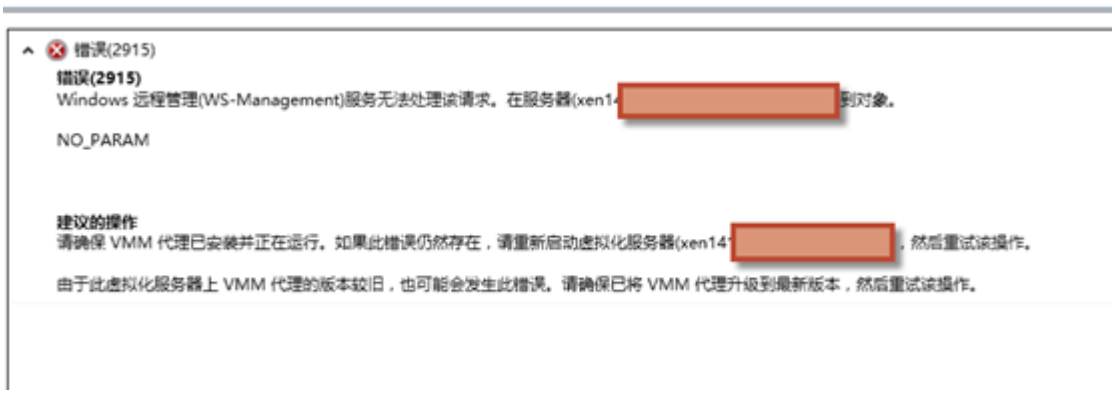
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\SSL 2.0

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\SSL 2.0\Client]

"DisabledByDefault"=dword:00000001



2915，一般是导入上一个注册表之后，再次添加 xenserver 造成的，主要原因是没有装 xenserver 的 SCVMM2012 组件，具体这个东西可以从思杰官方下载，全名叫什么可以参看我以前的博客。



22019，删除云，必须保证云里面空空如也才可以，这个算是对云的一个保护吧，不能算错误。

Http 协议 301、302 的原理和实现

作者：徐元振 来源：http://laoxu.blog.51cto.com/4120547/1410052

最近在配合其他团队对网站进行 seo 方面的优化，其中建议需要对 url 进行大量 301 修改，基本就是将原来的较长的 url 重新定向到一个比较短的 url，提高对搜索引擎的友好程度,如果发现你的网页从一个很精简的 url 被定向到一个冗长的 url 上，可能是被劫持了，对于 google 比较智能的搜索引擎，它还会接受之前的那个精简的，因为它更加友好化。今天就把这个 http 的 301 返回码分析一下。

HTTP 协议 301 返回码：简单的说就是永久重定向(Permanently Moved)

HTTP 协议 302 返回码：简单的说就是暂时重定向(Temporarily Moved)

实现 301、302 的重定向其实就是通过对 http 协议 location 的修改

用 php 的 header 函数去实现这个请求

```
<?php
header("HTTP/1.1 301 Moved Permanently");
header("Location: http://www.baidu.com/");
?>
```

可以看到我做个 301 永久重定向到 baidu 首页。

如果代码是这样呢？

```
<?php
header("Location: http://www.baidu.com/");
?>
```

发现如果 php 在 header 函数实现重定向方法时，不标明返回码为 301 默认是 302 暂时性重定向。

以上是在应用程序上实现 301 或者 302 跳转，明白了 301、302 重定向的原理是对 http 报文的 location 进行修改，但是我们一般都是在 web 服务器上去做。nginx 有一个 location 指令，它是不是可以修改 http 报文的 location 进行重新定义呢？

首先对一个静态页面访问，查看请求的头部信息

返回码是 200，并没有之前的请求报文的 location 标签信息

那么在 nginx.conf 配置文件里增加一段配置，意思是当请求以 html 结尾的文件重定向到对应的以 php 结尾的文件，也就是请求 index.html 重定向对应到 index.php 上

```
location ~ \.html$ {
    rewrite ^(.*)\.html$ $1.php permanent;
}
```

平滑重新启动 nginx 服务，再请求刚才的 url，返回码为 301，header 头部信息里增加了 location 信息，指明了被重新定向到 index.php 上

```
[root@localhost html]# curl -I http://192.168.39.139/index.html
HTTP/1.1 301 Moved Permanently
Server: nginx/1.6.0
Date: Sun, 11 May 2014 22:46:24 GMT
Content-Type: text/html
Content-Length: 184
Location: http://192.168.39.139/index.php
Connection: keep-alive
```

这就是 nginx 如何实现 301 的，nginx 实现 302 重定向，只要把 permanent 改成 redirect 即可

```
location ~ \.html$ {
    rewrite ^(.*)\.html$ $1.php redirect;
}
```

平滑重新启动 nginx 服务，http 的头部信息里面变成了 302 重定向

```
[root@localhost html]# curl -I http://192.168.39.139/index.html
HTTP/1.1 302 Moved Temporarily
Server: nginx/1.6.0
Date: Sun, 11 May 2014 22:55:38 GMT
Content-Type: text/html
Content-Length: 160
Location: http://192.168.39.139/index.php
Connection: keep-alive
```

注意避免 301 跳转的死循环，以下我同时在 nginx 和应用程序上实现了 301，也就是从 html 跳转到 php，然后又从 php 跳回 html 页面，出现这种情况 chrome 浏览器会出现这种提示

如果下次遇到这种问题，可能是出现重定向的死循环了。

IT 项目中存储设备的选型

作者：孙杰 来源：http://xjsunjie.blog.51cto.com/999372/1410019

在 IT 项目中，由于存储设备涉及到的投资规模相对较大，使用周期较长，因此它的重要性也就更高一些。那么在 IT 项目中如何进行存储设备的选型呢？下面我们就从以下几个方面做一个分析和说明。

选择合适的存储设备时，有几个因素必须要考虑进去，它们是：协议、容量、性能、可扩展性、易管理性和成本。除了这些标准的决定因素之外，你还要考虑所选择的存储设备不会处于网络、计算和监控管理平台之外的真空范围内。存储设备必须尽可能地与 IT 基础架构设施无缝整合在一起。

一、协议

协议是选择存储设备首先要考虑的问题。选择什么样的存储协议在 IT 项目整体决策中有很重要的作用，因为它可以而且还将继续决定使用哪一家厂商的产品和平台。另外，它还影响着基础设施的整体设计、体系结构和类型。光纤通道存储网络是目前企业数据中心使用率最高的存储网络，FC 协议和 iSCSI 协议发挥着重要的作用。FC 协议其实并不能翻译成光纤协议，只是 FC 协议普遍采用光纤作为传输线缆而不是铜缆，因此很多人把 FC 称为光纤通道协议。在逻辑上，我们可以将 FC 看作是一种用于构造高性能信息传输的、双向的、点对点的串行数据通道。在物理上，FC 是一到多对应的点对点的互连链路，每条链路终结于一个端口或转发器。FC 的链路介质可以是光纤、双绞线或同轴电缆。

iSCSI 技术是一种由 IBM 公司研究开发的，是一个供硬件设备使用的可以在 IP 协议的上层运行的 SCSI 指令集，这种指令集合可以实现在 IP 网络上运行 SCSI 协议，使其能够在诸如高速千兆以太网上进行路由选择。iSCSI 技术是一种新储存技术，该技术是将现有 SCSI 接口与以太网络(Ethernet)技术结合，使服务器可与使用 IP 网络的储存装置互相交换资料。而且 ISCSI 协议降低了系统的部署成本并解决了兼容性和统一管理问题，则代表了存储发展的未来。

另外，以太网光纤通道(FCoE)标准已经成型，产品推出市场已有数年的时间，端到端系统也已经成熟。FCoE 技术标准可以将光纤通道映射到以太网，可以将光纤通道信息插入以太网信息包内，从而让服务器-SAN 存储设备的光纤通道请求和数据可以通过以太网连接来传输，而无需专门的光纤通道结构，从而可以在以太网上传输 SAN 数据。FCoE 允许在一根通信线缆上传输 LAN 和 FC SAN 通信，融合网络可以支持 LAN 和 SAN 数据类型，减少数据中心设备和线缆数量，同时降低供电和制冷负载，收敛成一个统一的网络后，需要支持的点也跟着减少了，有助于降低管理负担。作为一个行业，由于网络合并优势多多，因此现在主要向以太网或 IP 协议(iSCSI 和 FCoE)的方向发展，但是光纤通道仍然占有较大的市场份额，因为它在传统投资、性能可靠性以及客户信任方面仍占有很大优势。一般来说需要结合网络存储结构（DAS、NAS 、SAN），来考虑整个项目的存储架构。

存储协议的比较

	SCSI协议	FC协议	iSCSI协议	AOE协议	Infiniband协议
接口技术	SCSI	光纤通道	IP	IP	Infiniband
接口类型	并行	串行	串行	并行	串行
适配器	SCSI卡	FC HBA	iSCSI HBA 或以太网卡	以太网卡	HCA(主机端） TCA（目标端）
目前最大速率	320MB/s	4Gb/s	10Gb/s	1Gb/s	60Gb/s
管理	简单	复杂	简单	简单	简单
兼容性	好	差	好	好	较好

二、容量与性能

容量和性能也是客户考虑的重要因素。传统的存储系统的最大磁盘容量由它们支持的磁盘数量和那些磁盘的大小决定。也就是说，最大容量通常会令客户感到困惑，因为没有分级和高速缓存等功能的

话，性能将于磁盘数量挂钩。因此你必须配备更多磁盘来保证 IOPS 性能，而且还要配备很多没有被使用的备用存储容量。

你想选择一个存储平台，这个平台可以在基础构架的工作周期中灵活扩展以满足你的容量需求。比较好的方法就是选择由若干个“通用存储设备”组成的用于存储的集群，组成集群存储的每个存储系统的性能和容量均可通过“集群”的方式得以叠加和扩展。当然即使使用了存储集群，一旦遇到存储系统的瓶颈，还会有两种选择：一是：采用硬件更加强大的单个存储系统；二是：采用若干个普通性能的存储系统来组成“存储的集群”。这样就可提供按比例增加的存储资源的性能、容量、可靠性及可用性，突破了单机设备的种种限制。

性能表现对于任何存储平台来说都意义重大。在一个典型的私有云部署平台中，在性能表现上被划分为多个层次，一般情况下可以被分为银级、金级和铂金级。你想让你的存储平台可以在不耗尽资源的情况下有多个服务层次(也就是说，你不想在每次服务中都为顶层服务花费)。存储的一些功能像自动分层处理(阵列频繁地从磁盘中读取数据，或者反过来)，电脑高速缓冲处理(前端的高速缓冲内存条)，可以满足在低消耗下实现高性能。此外，像数据去重、复制和快照等功能也是值得考虑的方面。

三、可扩展性、易管理性和成本

可扩展性

一般集群存储应该包括存储节点、前端网络、后端网络等三个构成元素，每个元素都可以非常容易地采用业界最新技术而不用改变集群存储的架构，且扩展起来非常方便，像搭积木一样进行存储的扩展。特别是对于那些对数据增长趋势较难预测的用户，可以先购买一部分存储，当有需求的时候，随时添加，而不会影响现有存储的使用。

分布式存储能提高系统的性能,尤其是可扩展性。所有对集群存储的操作都经由分布式操作系统统一

调度和分发，分散到集群存储各个存储节点上完成。使用分布式操作系统带来的好处是各节点之间没有任何区别，没有主次、功能上的区别，所有存储节点功能完全一致，这样才能真正做到性能最优。

易管理性

目前存储业界的管理方式都是通过各厂商的管理工具，或通过 Web 界面进行管理和配置，往往客户端还需要安装相关软件才能访问到存储上的空间。随着需要管理的存储空间逐渐增大，管理存储的复杂度和管理人员的数量也将会随之增加。而集群存储应该提供一种集中的、简便易用的管理方式，对客户端没有任何影响，采用业界标准的访问协议（比如 NFS，CIFS）访问集群存储。

成本

成本在任何 IT 项目中都是一个很关键的因素。对于私有云来说，你一定需要一个能被本地工具和你自己构建的自动化监控平台同时管理和检测的系统。存储管理平台越开放，它的应用程序界面或软件系统就越强大，使用起来就越方便。当你作出决定的时候，你一定要确保你为自动化监控管理平台选择的软件能够很好的适应和管理你的存储平台。这些工具会经常被使用并且价格昂贵，它们对你选择硬件的限制多不多就显得很重要。

总之，对于一个 IT 项目来说，存储设备的选择很复杂也很重要。即使你现在没有考虑虚拟化，为了以后存储平台在云方面的发展，你也要反复斟酌你的选择。并且在 IT 系统复杂度和风险不断增加、采购和管理成本不断上涨的情况下管理好存储和数据，对所有企业来说都是一个严峻的挑战。但概括说来，企业的存储需求大同小异，我们可以简要归纳为以下几点：1、总体拥有成本(TCO)有效控制；2、数据的安全、可靠性、一致性；3、以低成本提供杰出的高端存储服务；4、实现不同存储级的整合；5、降低存储管理复杂性；6、存储可扩展性和高性能；7、数据存储生命周期管理。把握好了这几点，在进行存储设备选型时还是对我们有帮助的。

用 VMware Fusion 来“穿越”

作者：陈巨廉 来源：http://2574478.blog.51cto.com/2564478/1408482

一. 背景介绍

曾经听到过这样一个比喻，说的是在企业里，喜欢用 Windows 操作系统的是普通型员工；喜欢用 Linux 操作系统的是宅男型员工；而喜欢用 MacOS 操作系统的是文艺型员工。我所支持的是一家以为各个企业定制书籍的文化传媒公司。公司以《论语》为企业精神，可想而知整个企业里都充斥着“文艺型销售员工”。2013 年该公司筹划归并入国外某知名品牌公司，除了服务器端的迁移外，很多国内员工的电脑操作系统及其应用软件也要从各自为政的 MacOS 上并入国外统一管理的 Windows 上。有变革就有阵痛，对于我们 IT 人员来说一面是共事多年且身处公司盈利部门的老同事，一面是中规中矩的标准化新策略，如何通过“checkand balance”来找到平衡点是我们的首要任务。

通过几个月的研究和测试，我们最终选用 VMware 的 Fusion 产品族很好的解决了这个两难的局面。

二. 需求分析

一开始，通过问卷与访谈的方式，我们逐一排查，了解并收集到了如下现状与需求：

1. 在 2013 年中，就在合并前夕，公司为国内各个办公室的高级销售人员和管理层配备了 Macair 笔记本电脑，加上员工自带和初始申请的 AppleMAC，合计 MAC 电脑的普及度达到约 75%，且型号并不一致。

2. 从操作习惯和认可度方面，很难说服现有员工摒弃使用习惯了的 Mac 平台改用英文版的 Windows 操作系统。

3. 全部更换电脑成本太高，而让 IT 人员逐一进行用户数据备份、迁移以及重装操作系统的话，由于外勤人员回到公司的时间不定且次数有限，无法给 IT 人员充沛的时间进行操作。何况在整体归并期间 IT 部门的主要精力和时间应当花费在服务器端的迁移和测试上。

4. 由于网上银行控件只能在 Windows 下运行的原因，MacOS 的用户一直以来都抱怨用非 IE 浏览器无法用某些国内银行的网上银行进行支付等操作。

5. 国外总部有很多内网应用程序以及 web 环境都是以 Windows 及其 IE 平台开发的。

6. 销售人员经常碰到与客户往来的文件资料是基于 Windows 的各种软件所处理过的。

7. 用户显然无法接受过于技术复杂度的“双系统”的方法，何况“双系统”无法满足同时两种系统同时并行且互通互联的需求。

三. “穿越”计划

基于上述原因，IT 团队秉承着一直以来“死磕自己，方便大家”的精神，经过技术研究和产品选型最终定位使用 VMwareFusion 来实现用户端操作系统的快速归并和无缝延续。基本思路就是：IT 部门定制好一个全新的 Windows 操作系统的 vmdk 文件，通过网络下载或硬盘拷贝到用户 Mac 电脑，利用 VMwareFusion 快速加载到用户的 MacOS 桌面上。由于其易操作性，用户很容易上手并能按需启用 Windows。后继，IT 人员只需要对那个 vmdk 文件进行更新、维护、修复、升级等操作，从而把对用户 Mac 电脑的影响降到最低。具体好处罗列如下：

1. VMware Fusion 可确保 Windows 及其应用程序在 Mac 电脑上无痕运行，且不占用过多系统资源。

2. 通过 VMware Fusion，MacOS 可以启动 Windows 实现文件的相互共享，方便调用和操作。用户可以在两个系统间无缝复制、粘贴和拖放各种文件。双系统的网络连接互通，打印设备也可共享。另外 Windows 还可读取插在 Mac 电脑上的 USB 移动存储设备。

3. 在用户不需要的时候，可以将 Windows 应用程序最小化至 Mac OS 桌面的 Dock 中。

4. 和总部或者客户开视频会议时，Windows 可以调用 Mac 上的 iSight 摄像头。

5. VMware Fusion AutoProtect 会自动定期地拍摄快照，保存用户 Windows 的一个状态，当其 Windows 系统崩溃，中毒或者受损的时候，用户只需几步简单操作便可自行恢复到此前的某个系统健康状态。

6. 对于一些 VIP 用户，在条件允许的情况下，Converter 可以把其原先使用的 Windows 系统迁移成虚拟机。再通过集成的“迁移助理”加载到新的 Mac 上，从而节约并回收设备成本。

7. 通过 VMwareFusion 的工具，Windows 的各种应用甚至可以沿用到双或多显示器的用户电脑上。

四. “穿越”实施

由于涉及到公司信息，我们暂时用测试环境演示穿越过程。

1. 准备软件环境

通过 VMware 系列软件，先由总部 IT 创建一个最新的且符合总部各种软件标准以及规范的 Windows 的 vmdk 文件。

在用户装有 Mac OS 的电脑上，双击安装 WMwareFusion 6 的 dmg 文件，并对其进行性能调优。考虑到用户电脑的硬件配置不同以及真实使用到 Windows 的可能性，我们默认是给虚拟机分配“1 个处理器核心”和“2048M”的内存（如下图所示），同时配置网卡并开启共享。

通过网络或移动硬盘将含有 Windows 的 vmdk 文件拷贝到用户电脑上。

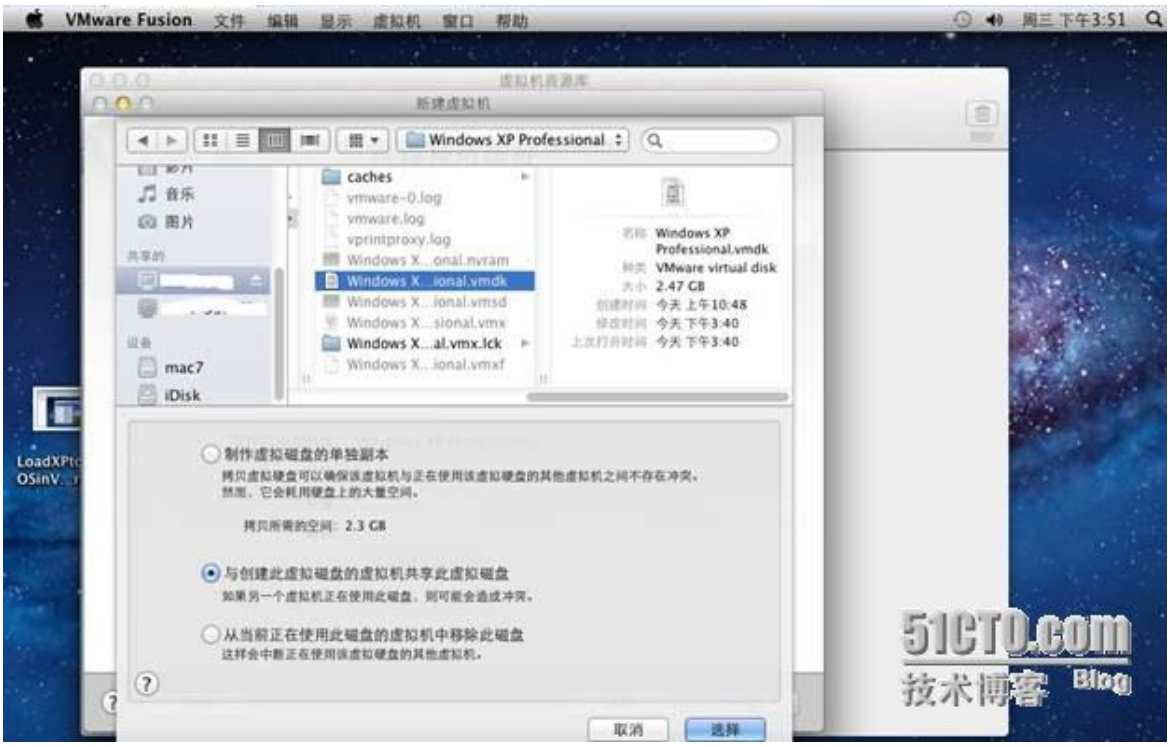
2. 安装并加载 vmdk

运行 VMware Fusion，在“更多选项...”里选择“创建自定义虚拟机”（如下图所示）。

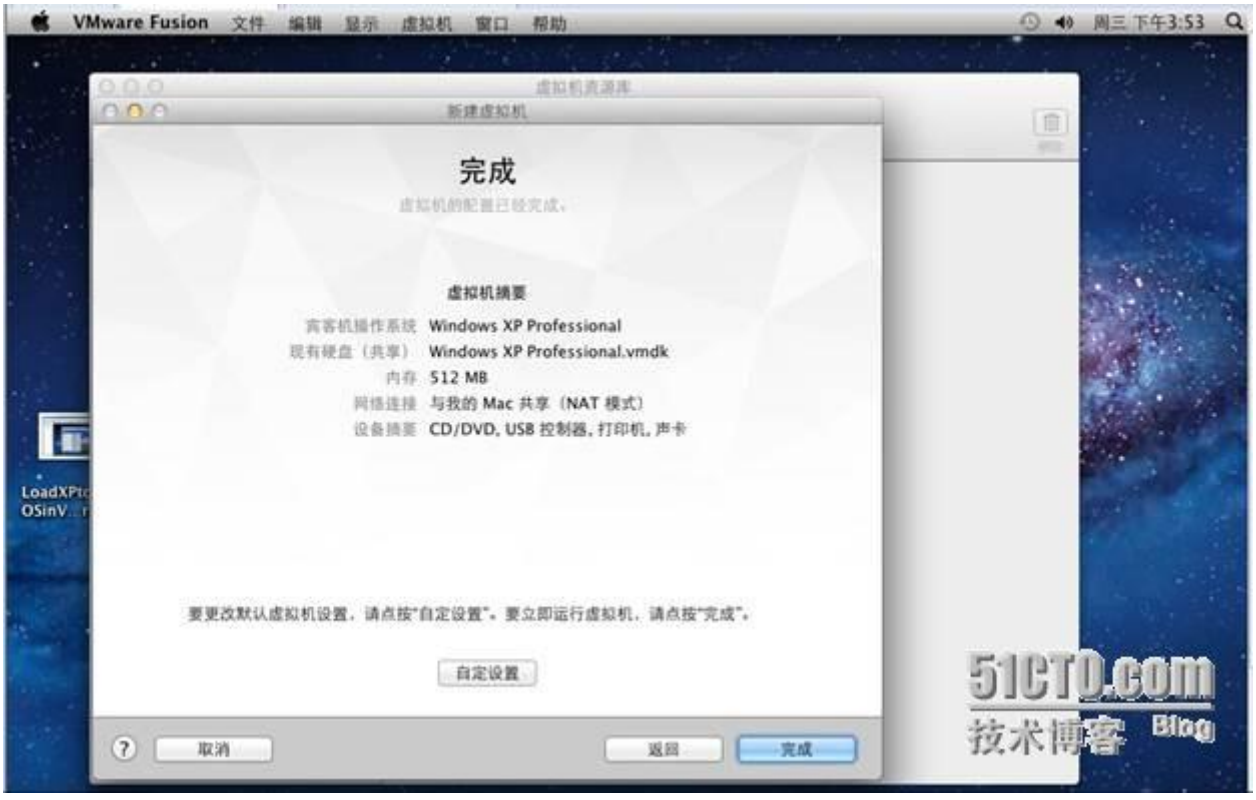
选择操作和 vmdk 文件相匹配的 Windows 操作系统，这里是测试环境所以选用的是 WindowsXP Professional（如下图所示）。

选择使用“选择虚拟磁盘...”（如下图所示）。

选择 vmdk 文件所在的路径，并点选“与创建此虚拟磁盘的虚拟机共享此虚拟磁盘”（如下图所示）。



点击“完成”，Fusion 将自动开始加载（如下图所示）。



整体加载完成后，就可以看到 Windows 的系统界面了（如下图所示）。

为了尽量不影响用户的 Mac 日常使用，我们把 Windows 设置成了 Unitymode（如下图所示），另外，用户不但可以从 dock 上启动 Windows 应用，还能在 Spotlight 上触发。

3.快照备份

考虑到备份文件的大小，在加载好 Windows 系统后，立即关机并对其进行第一次“拍照”（如下图所示）。同时对“AutoProtect”进行拍摄间隔和快照数量的设置。

五.“穿越”心得

安装了 VMwareFusion 的 Mac 电脑可谓是如虎添翼，用户轻松的在 MacOS 和 Windows 间灵活“穿越”，Mac 电脑在功能上有了进一步的扩展，企业归并进程得以大幅加快，IT 人员在用户（特别是外勤用户）电脑上花费的时间和精力显著减少，设备更换成本明细减低。更重要的是用户的认可度非常高。

苹果系统 Sequel Pro—MySQL 客户端工具一个大坑

作者：贺春旸 来源：http://hcymysql.blog.51cto.com/5223301/1411287

软件名称:Sequel Pro（MySQL 客户端工具）

官网地址:http://www.sequelpro.com/

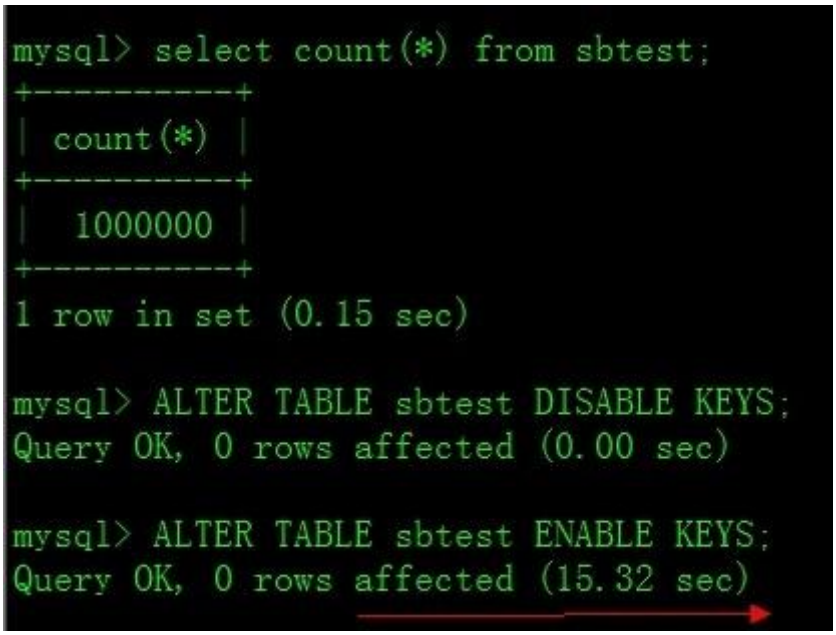
该工具在导出表数据的时候，会产生一条坑爹的 SQL：

```
/*!40000 ALTER TABLE `sbtest` DISABLE KEYS */;
```

这条 SQL 是针对 MyISAM 引擎批量插入，为了加快插入速度，会加上 ALTER TABLE t1 DISABLE KEYS;来关闭索引， 这里存在一定的风险，因为在插入完毕后，要再执行 ALTER TABLE t1 ENABLE KEYS;来开启索引，这里开启索引相当于索引重建了，非常耗时，而且会锁表。应禁止使用该功能。

而 InnoDB 引擎因采用内存型缓冲写，设计上不同于 MyISAM 引擎，对此不受影响。

下面是一个测试，截图如下：



如果你有一张很大的 MyISAM 表，会非常耗费时间，在开启索引的时候，会全表锁，坑死你没商量。

知识回顾：

MySQL 注释符有三种：

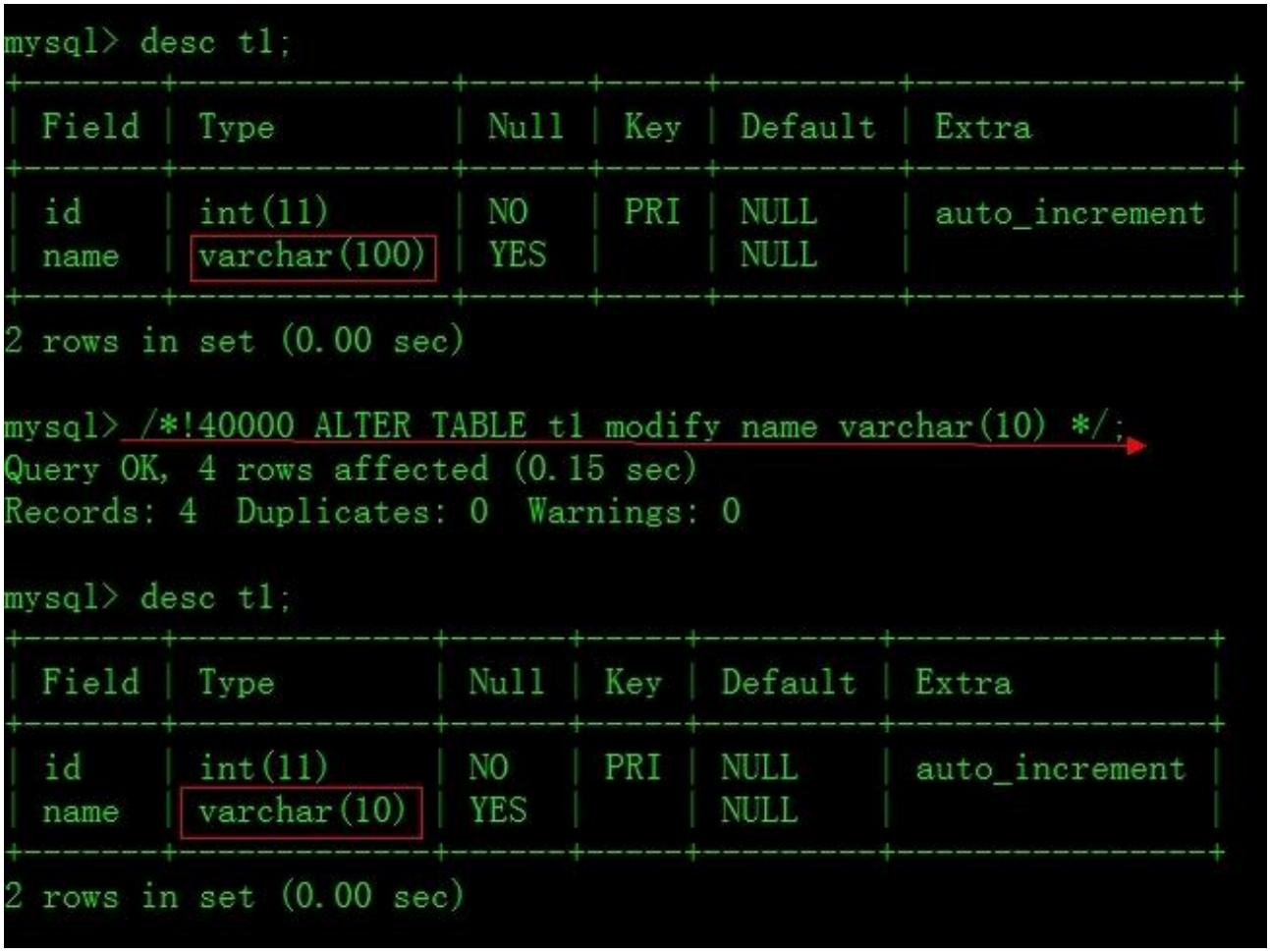
1、#...

2、"-- ..." （注：“--”错了，需要一个空格,应该是“-- ”。）

3、/*...*/ （注释多行）

而/*!40000 ..... */并非注释，意思为 MySQL4.00.00 版本以上可以执行，低于则不能执行。

例：



使用苹果笔记本的同学，可得小心了，如果哪天你看走了眼，没有把**/*!40000 ALTER TABLE t1 DISABLE KEYS;*/给删掉**，且你导入数据的表还是 MyISAM 引擎，那么我相信，你可以把你的苹果笔记本给砸了。

IM 系统架构设计之浅见

作者：yaocoder 来源：http://yaocoder.blog.51cto.com/2668309/1412029

背景：除去大名鼎鼎的 QQ 这款即时聊天工具，还有许多细分行业的 IM，比如淘宝阿里旺旺、网易泡泡、YY 语音……。恰巧公司产品也要开发一款基于我们自己行业的类 IM 系统，很有幸我担当了这个产品的架构师，核心代码编写、实现者。下面我近年来从技术上我对 IM 系统（即时消息的传输，不包括语音，视频，文件的传输）的理解和设计分享出来，浅薄之见，望大家别见笑，欢迎给出批评意见。

一． 网络传输协议的选择

目前我知晓的所有 IM 系统传输即时消息无外乎使用 UDP、TCP、基于 TCP 的 http 这几种协议中的一种或几种。比如 QQ 主要采用 UDP 协议，MSN 主要采用 TCP 协议，而且他们也都支持 HTTP 协议的代理模式。更多资料，请参加这篇文章《一些常用软件的网络端口协议分类介绍》。

我们该如何选择呢？

- UDP 协议实时性更好，但是如何处理安全可靠的传输并且处理不同客户端之间的消息交互是个难题，实现起来过于复杂；
- HTTP 协议属于扩展支持，我们在产品的初始阶段可以不用支持；
- 那就非 TCP 协议莫属了，要考虑的同样也有很多，特别是如果有海量用户的需求。如何保证单机服务器高并发量，如何做到灵活，扩展的架构。

> Tips: QQ 为什么采用 UDP 协议，而不采用 TCP 协议实现？

二. 应该选择什么格式的数据协议

二进制格式？文本格式？这个话题转到我的这篇文章《网络传输数据格式的选择》，从我们当前的需求和产品周期上我觉得选择 JSON 形式的数据协议是最好的。

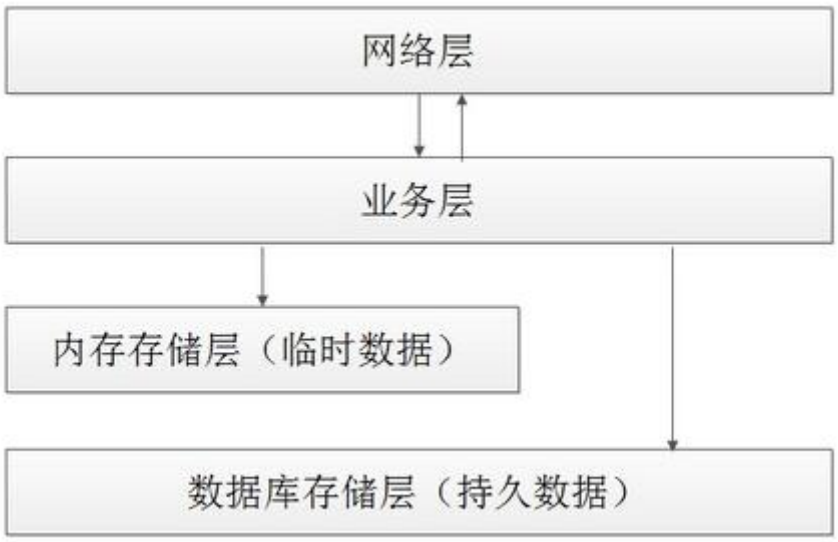
三. 架构设计

首先我们来提炼一下一个 IM 系统的主要需求，包括账号，关系链，在线状态显示，消息交互……。

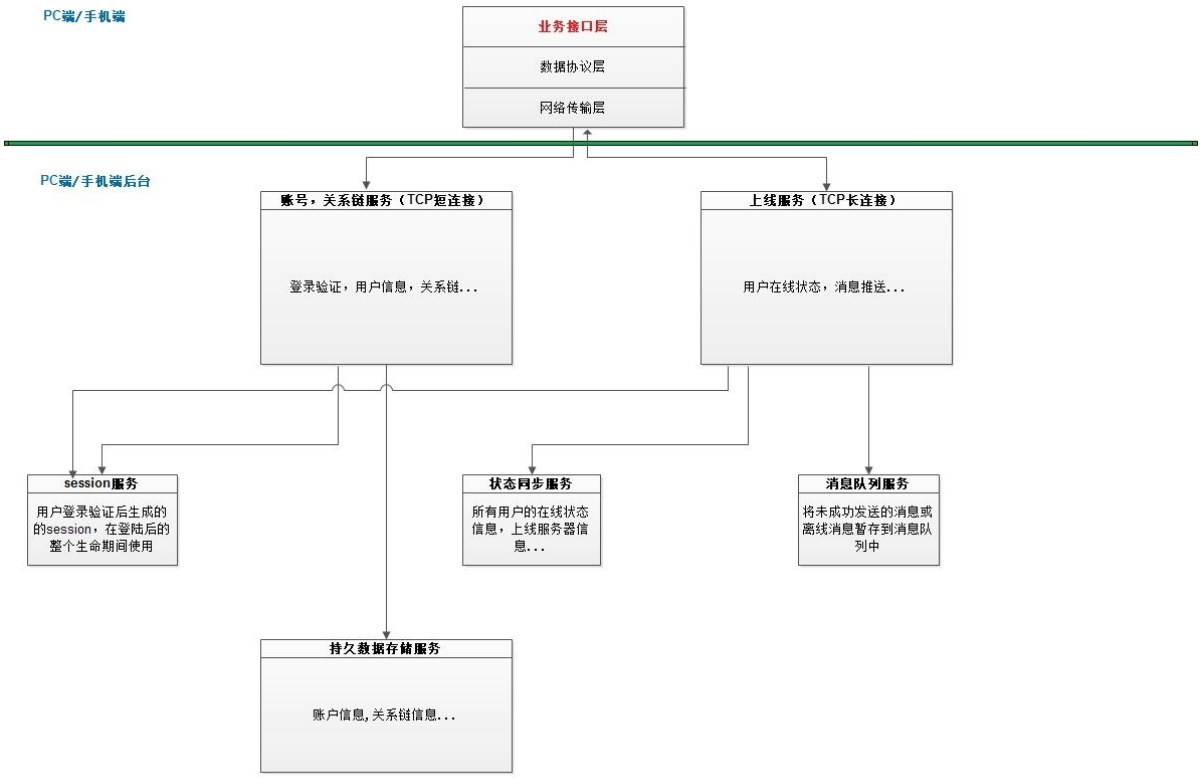
架构考量：

- 由于采用可靠传输协议 TCP，考虑到负载问题（短连接实现账号、关系链相关业务，长连接实现上线、信息推送）；
- 后台架构的灵活性、可扩展性，支持分布式部署——把网络层、业务逻辑层、数据层分离，网络层和业务层支持负载均衡策略、数据层支持分布式存储；
- 客户端 SDK 的易用性：把网络层、数据层分离、业务逻辑层分离；

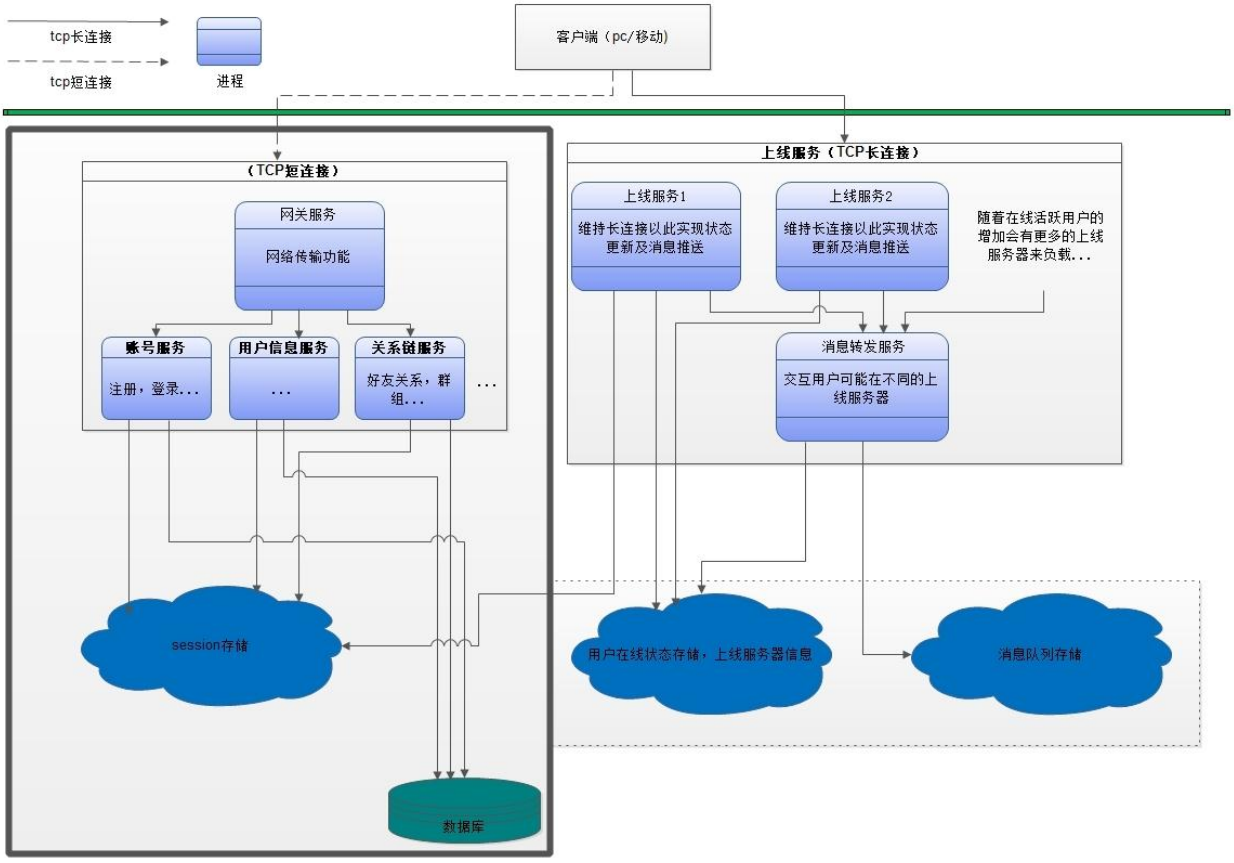
后台架构简化图



架构示意图


PC端/手机端
业务接口层
数据协议层
网络传输层
PC端/手机端后台
账号，关系链服务（TCP短连接）
登录验证，用户信息，关系链...
上线服务（TCP长连接）
用户在线状态，消息推送...
session服务
用户登录验证后生成的的session，在登陆后的整个生命期间使用
状态同步服务
所有用户的在线状态信息，上线服务器信息...
消息队列服务
将未成功发送的消息或离线消息暂存到消息队列中
持久数据存储服务
账户信息,关系链信息...

架构细化图


tcp长连接
tcp短连接
进程
客户端（pc/移动）
（TCP短连接）
网关服务
网络传输功能
账号服务
注册，登录...
用户信息服务
...
关系链服务
好友关系，群组...
...
上线服务（TCP长连接）
上线服务1
维持长连接以此实现状态更新及消息推送
上线服务2
维持长连接以此实现状态更新及消息推送
随着在线活跃用户的增加会有更多的上线服务器来负载...
消息转发服务
交互用户可能在不同的上线服务器
session存储
用户在线状态存储，上线服务器信息
消息队列存储
数据库

说明

- 从<架构细化图>中可以看出对于上线服务由于建立的是 TCP 长连接，对于单台服务器往往由于硬件资源、系统资源、网络资源的限制无法做到海量用户的同时在线，所以设计为根据服务器负载支持多服务器上线，同时由于多服务器上线造成了对整个系统交互（不同的客户端的交互，协作部门应用服务和客户的交互）的分割，引入消息转发服务器作为粘合点。另外对于多服务器上线造成的统一账户信息（在线状态，消息）数据的分割，引入统一的数据层（内存存储层：session、状态信息存储、消息队列存储；数据库：账号信息存储）做到业务和数据的分离，也就做到了支持分布式部署。参见我的这篇文章《构建高性能服务的考量》

- 对于部分业务服务：做到网络层、业务层、数据层的完全分离。首先对于 TCP 短连接来说不会如长连接那般消耗资源，即使后期遇到海量的并发访问请求依然可以从容的通过负载均衡策略和数据分布式部署策略进行解决。参见我的这篇文章《服务端架构中的“网关服务器”》

服务端平台及技术选型

- 系统开发平台： CentOS——Linux 发行版的一种，稳定可靠、可定制优化、支持丰富；
- 网络支撑层： libevent——减小开发成本，增强稳定性；
- 缓存存储层： Redis——支持丰富的存储结构，支持分布式存储；
- 数据库： MySQL——最适合互联网的数据库，免授权、高效稳定、可控性高；
- 开发语言： C/C++；

部分热点问题考量

➢ 系统性能考量：

- 编码角度：采用高效的网络模型，线程模型，I/O 处理模型，合理的数据库设计和操作语句的优化；
- 垂直扩展：通过提高单服务器的硬件资源或者网络资源来提高性能；
- 水平扩展：通过合理的架构设计和运维方面的负载均衡策略将负载分担，有效提高性能；后期甚至可以考虑加入数据缓存层，突破 IO 瓶颈；

➢ 系统的高可用性：（防止单点故障）

- 在架构设计时做到业务处理和数据的分离，从而依赖分布式的部署使得在单点故障时能保证系统可用。
- 对于关键独立节点可以采用双机热备技术进行切换。
- 数据库数据的安全性可以通过磁盘阵列的冗余配置和主备数据库来解决。

Percona5.6 首次提供了审计日志功能

作者：贺春旸 来源：http://hcymysql.blog.51cto.com/5223301/1413670

如果有一天数据库里丢失了一条记录，开发让你查什么时候被谁从哪个 IP，执行了 delete 操作，那么在之前的版本是无法查到的，而 Percona 最新版本 5.6.17 首次提供了审计日志功能。

（注：截止 2014 年 5 月 19 日，官方 MySQL5.6.17 社区版和 MariaDB10.0.11 均没有提供该功能）

版本：

```
5.6.17-65.0-rel65.0-log Percona Server with XtraDB (GPL), Release rel65.0, Revision 587
```

安装插件：

```
INSTALL PLUGIN audit_log SONAME 'audit_log.so';
```

show plugins;你会发现

```
| audit_log                     | ACTIVE   | AUDIT                | audit_log.so | GPL     |
```

变量参数：

```
mysql> show global variables like 'audit%';
+-------------------------+-------------+
| Variable_name           | Value       |
+-------------------------+-------------+
| audit_log_buffer_size   | 1048576     |
```

```
| audit_log_file                | audit.log      |
| audit_log_flush               | OFF            |
| audit_log_format              | NEW            |
| audit_log_policy              | ALL            |
| audit_log_rotate_on_size | 0                |
| audit_log_rotations           | 0              |
| audit_log_strategy            | ASYNCHRONOUS |
+-------------------------+--------------+
8 rows in set (0.00 sec)
```

它先会把日志写入内存，然后再刷入磁盘里。

audit_log_strategy 参数可以调整下面 4 个策略

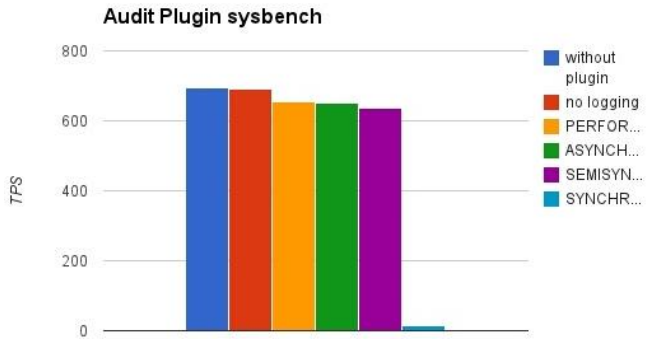
1、ASYNCHRONOUS log using memory buffer, do not drop events if buffer is full

2、PERFORMANCE log using memory buffer, drop events if buffer is full

3、SEMISYNCHRONOUS log directly to file, do not fsync every event

4、SYNCHRONOUS log directly to file, fsync every event

4 种策略性能压力测试：

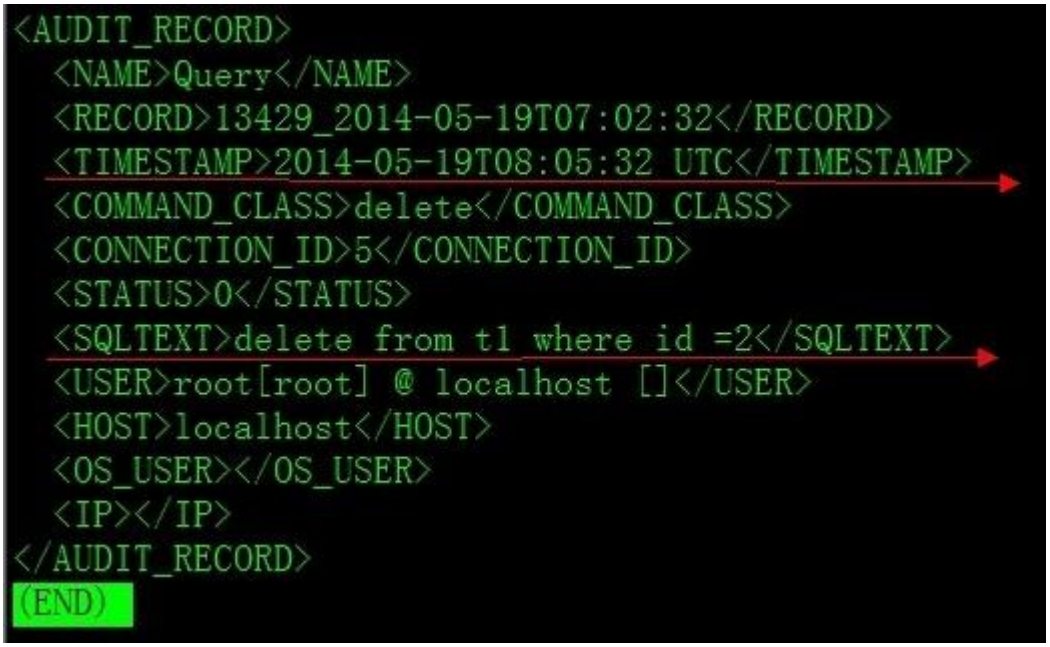


这里用默认就可以。

audit_log_rotate_on_size 参数表示超过定义的值，会自动轮训，切分日志。

audit_log_rotations 参数限制表示文件的数量。

效果：



MySQL 和 PostgreSQL 导入数据对比

作者：杨涛涛 来源：http://yueliangdao0608.blog.51cto.com/397025/1413445

在虚拟机上测评了下 MySQL 和 PostgreSQL 的各种 LOAD FILE 方式以及时间。 因为是虚拟机上的测评，所以时间只做参考，不要太较真，看看就好了。

MySQL 工具：

1. 自带 mysqlimport 工具。
2. 命令行 load data infile ...
3. 利用 mysql-connector-python Driver 来写的脚本。

PostgreSQL 工具：

1. pgloader 第三方工具。
2. 命令行 copy ... from ...
3. 利用 psycopg2 写的 python 脚本。

测试表结构：

```
mysql> desc t1;
+----------+-----------+------+-----+------------------+-------+
| Field    | Type      | Null | Key | Default          | Extra |
+----------+-----------+------+-----+------------------+-------+
| id       | int(11)   | NO   | PRI | NULL             |       |
```

```
| rank        | int(11)     | NO     |        | NULL                    |          |
| log_time | timestamp | YES    |        | CURRENT_TIMESTAMP |          |
+----------+-----------+------+-----+-------------------+-------+
3 rows in set (0.00 sec)
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|   1000000 |
+----------+
1 row in set (6.80 sec)
```

测试 CSV 文件：

t1.csv

MySQL 自身的 loader: （时间 24 妙）

```
mysql> load data infile '/tmp/t1.csv' into table t1 fields terminated by ',' enclosed by '"' lines
terminated by '\r\n';
Query OK, 1000000 rows affected (24.21 sec)
Records: 1000000    Deleted: 0    Skipped: 0    Warnings: 0
```

MySQL python 脚本：（时间 23 秒）

```
>>>
```

Running 23.289 Seconds

MySQL 自带 mysqlimport：（时间 23 秒）

```
[root@mysql56-master ~]# time mysqlimport t_girl '/tmp/t1.csv' --fields-terminated-by=',' --
fields-enclosed-by='"' --lines-terminated-by='\r\n' --use-threads=2 -uroot -proot
t_girl.t1: Records: 1000000  Deleted: 0  Skipped: 0  Warnings: 0
real    0m23.664s
user    0m0.016s
sys     0m0.037s
```

PostgreSQL 自身 COPY：（时间 7 秒）

```
t_girl=# copy t1 from '/tmp/t1.csv' with delimiter ',';
COPY 1000000
Time: 7700.332 ms
```

Psycopg2 驱动 copy_to 方法：（时间 6 秒）

```
[root@postgresql-instance scripts]# python load_data.py
Running 5.969 Seconds.
```

Pgloader 导入 CSV：（时间 33 秒）

```
[root@postgresql-instance ytt]# pgloader commands.load
                table name       read   imported   errors            time
                  ytt.t1      1000000    1000000        0          33.514s
```

```
----------------------------  ---------  ---------  ---------  --------------

----------------------------  ---------  ---------  ---------  --------------

            Total import time      1000000      1000000            0            33.514s
```

Pgloader 直接从 MySQL 拉数据：（时间 51 秒）

```
[root@postgresql-instance ytt]# pgloader commands.mysql

                  table name          read     imported       errors                time

               fetch meta data             2            2            0             0.138s

----------------------------  ---------  ---------  ---------  --------------

                           t1      1000000      1000000            0            51.136s

----------------------------  ---------  ---------  ---------  --------------

----------------------------  ---------  ---------  ---------  --------------

----------------------------  ---------  ---------  ---------  --------------

            Total import time      1000000      1000000            0            51.274s
```

附上 commands.load 和 commands.mysql

```
commands.load:

LOAD CSV

FROM '/tmp/ytt.csv' WITH ENCODING UTF-8

     (

          id, rank, log_time

     )

INTO postgresql://t_girl:t_girl@127.0.0.1:5432/t_girl?ytt.t1

WITH skip header = 0,
```

```
      fields optionally enclosed by '"',

      fields escaped by backslash-quote,

      fields terminated by ','

SET work_mem to '32 MB', maintenance_work_mem to '64 MB';

commands.mysql:

LOAD DATABASE

 FROM mysql://python_user:python_user@192.168.1.131:3306/t_girl?t1

 INTO postgresql://t_girl:t_girl@127.0.0.1:5432/t_girl?ytt.t1

 with data only

SET maintenance_work_mem to '64MB',

   work_mem to '3MB',

   search_path to 'ytt';
```

附 pgloader 手册：

http://pgloader.io/howto/pgloader.1.html

在 python 下比 celery 更加简单的异步任务队列 RQ

作者：芮峰云 来源：http://rfyiamcool.blog.51cto.com/1030776/1411358

前言：

这里介绍一个 python 下，比 celery 更加简单的异步工具，真的是很简单，当然他的功能没有 celery 多，复杂程度也没有 celery 大，文档貌似也没有 celery 多，但是为啥会介绍 rq 这个东西 因为他够简单。

当然他虽然简单，但是也是需要中间人的，也就是 Broker，这里只能是 redis 了。 他没有 celery 支持的那么多，比如 redis rabbitmq mongodb mysql 之类的。 说回来，咱们用 rq，就是看重他的简单。

如对 celery 有兴趣，可以看看我以前写过的博文。

http://rfyiamcool.blog.51cto.com/1030776/1325062

安装 redis 以及 python-rq 包，redis 的话，直接 yum 就行，python rq 需要 pip 来搞定。

```
[root@67 ~]# pip install rq
Downloading/unpacking rq
Downloading rq-0.4.5.tar.gz
Running setup.py egg_info for package rq
warning: no previously-included files matching '*' found under directory 'tests'
Requirement already satisfied (use --upgrade to upgrade): redis>=2.7.0 in
/usr/lib/python2.6/site-packages (from rq)
Downloading/unpacking importlib (from rq)
```

```
Downloading importlib-1.0.3.tar.bz2
Running setup.py egg_info for package importlib
Requirement already satisfied (use --upgrade to upgrade): argparse in /usr/lib/python2.6/site-packages (from rq)
Installing collected packages: rq, importlib
Running setup.py install for rq
warning: no previously-included files matching '*' found under directory 'tests'
Installing rqinfo script to /usr/bin
Installing rqworker script to /usr/bin
Running setup.py install for importlib
Successfully installed rq importlib
Cleaning up...
```

先开始官方的 demo：

这个是咱们要后端异步的模块：

```
import requests
def count_words_at_url(url):
resp = requests.get(url)
return len(resp.text.split())
```

创建队列

```
from redis import Redis
```

```
from rq import Queue

q = Queue(connection=Redis())
```

然后，直接 rqworker ！

一直往队列里面扔任务。

```
In [238]: result = q.enqueue(
              count_words_at_url, 'http://nvie.com'
)
In [241]: result = q.enqueue(
              count_words_at_url, 'http://nvie.com'
)
In [244]: result = q.enqueue(
              count_words_at_url, 'http://nvie.com'
)
In [247]: result = q.enqueue(
              count_words_at_url, 'http://xiaorui.cc'
)
In [250]: result = q.enqueue(
              count_words_at_url, 'http://xiaorui.cc'
)
In [253]: result = q.enqueue(
              count_words_at_url, 'http://xiaorui.cc'
```

```
)
In [256]: result = q.enqueue(
              count_words_at_url, 'http://xiaorui.cc'
)
```

rqworker 的接口任务并执行：

（下面的 log 已经说明了一切，任务确实执行了，而且我在 ipython 下，很是流畅，我不需要担心任务是否很好的执行，我只需要把任务一扔，就甩屁股走人了。）

```
00:42:13 *** Listening on default...
00:42:22 default: nima.count_words_at_url('http://xiaorui.cc') (84f9d30f-8afc-4ea6-b281-4cb75c77779f)
00:42:22 Starting new HTTP connection (1): xiaorui.cc
00:42:23 Starting new HTTP connection (1): rfyiamcool.blog.51cto.com
00:42:23 Job OK, result = 2632
00:42:23 Result is kept for 500 seconds.
00:42:23
00:42:23 *** Listening on default...
00:42:27 default: nima.count_words_at_url('http://xiaorui.cc') (9fdaa934-e996-4719-8fb5-d619a4f15237)
00:42:27 Starting new HTTP connection (1): xiaorui.cc
00:42:28 Starting new HTTP connection (1): rfyiamcool.blog.51cto.com
00:42:28 Job OK, result = 2632
```

```
00:42:28 Result is kept for 500 seconds.

00:42:28

00:42:28 *** Listening on default...

00:42:28 default: nima.count_words_at_url('http://xiaorui.cc') (952cc12b-445e-4682-a12a-
96e8019bc4a8)

00:42:28 Starting new HTTP connection (1): xiaorui.cc

00:42:28 Starting new HTTP connection (1): rfyiamcool.blog.51cto.com

00:42:28 Job OK, result = 2632

00:42:28 Result is kept for 500 seconds.

00:42:28

00:42:28 *** Listening on default...

00:42:29 default: nima.count_words_at_url('http://xiaorui.cc') (c25803e4-a3ad-4889-bbec-
06cf1e77a11e)

00:42:29 Starting new HTTP connection (1): xiaorui.cc

00:42:29 Starting new HTTP connection (1): rfyiamcool.blog.51cto.com

00:42:29 Job OK, result = 2632

00:42:29 Result is kept for 500 seconds.

00:42:29

00:42:29 *** Listening on default..
```

紧接着咱们再跑一个我自己测试的模块，逻辑很简单在 sleep 情况下，是否会很好的执行，来测试他的异步任务执行。 当然你也可以 rqworker 执行的运行，下面的代码更像是 event 事件的感觉。

```
[root@67 ~]# cat worker.py
```

```
#xiaorui.cc

import os

import redis

from rq import Worker, Queue, Connection

listen = ['high', 'default', 'low']

redis_url = os.getenv('REDISTOGO_URL', 'redis://localhost:6379')

conn = redis.from_url(redis_url)

if __name__ == '__main__':

with Connection(conn):

    worker = Worker(map(Queue, listen))

    worker.work()
```

下面是自己需要异步执行的模块代码~

```
[root@67 ~]# cat utils.py

#xiaorui.cc

import requests

import time

def tosleep(num):

time.sleep(num)

return num
```

咱们在 ipython 测试下吧：

```
redis import Redis
```

```
In [54]: from rq import Queue

In [55]:

In [56]: q = Queue(connection=Redis())

In [57]: from utils import tosleep

In [58]: for i in range(5):

q.enqueue(tosleep,5)

    ....:

    ....:

Out[59]: Job(u'8d71a0ee-695a-4708-b6cf-15821aac7299',

enqueued_at=datetime.datetime(2014, 5, 14, 15, 42, 4, 47779))

Out[59]: Job(u'27419b10-8b12-418c-8af1-43c290fc2bf3',

enqueued_at=datetime.datetime(2014, 5, 14, 15, 42, 4, 51855))

Out[59]: Job(u'7c98f0d1-7317-4c61-8bfa-10e223033948',

enqueued_at=datetime.datetime(2014, 5, 14, 15, 42, 4, 53606))

Out[59]: Job(u'0a84a48f-3372-4ef0-8aa8-d868de2e0c11',

enqueued_at=datetime.datetime(2014, 5, 14, 15, 42, 4, 57173))

Out[59]: Job(u'ad1986b9-a2fa-4205-93ab-a1b685d7cf88',

enqueued_at=datetime.datetime(2014, 5, 14, 15, 42, 4, 58355))
```

看到没有，本来咱们调用了一个函数是 sleep5s，但他不影响其他的代码的堵塞，会扔到队列里面后，迅速的执行后面的代码。

如果我想像 celery 那样，查看结果的话，也是用 result 方法的。

```
#xiaorui.cc

In [67]: job=q.enqueue(tosleep,5)

In [68]: job.result

In [69]: job.result

In [70]: job.result

In [71]: job.result

In [72]: job.result

Out[72]: 5
```

但是有个缺点，任务是异步方式的放到了 redis 的队列里面了，但是后端的 work 貌似是单进程的。。。当然也很好改，用 threading 针对每个任务进行 fork 线程就可以了。

```
#xiaorui.cc

In [47]: for i in range(5):

    .....:         q.enqueue(tosleep,5)

    .....:

    .....:

Out[47]: Job(u'5edb3690-9260-4aba-9eaf-fa75fbf74a13',

enqueued_at=datetime.datetime(2014, 5, 14, 15, 24, 54, 229289))

Out[47]: Job(u'e91cfcb8-850b-4da4-8695-13f84a6a0222',

enqueued_at=datetime.datetime(2014, 5, 14, 15, 24, 54, 233016))

Out[47]: Job(u'cc6c78d4-e3b5-4c22-b027-8c070b6c43db',

enqueued_at=datetime.datetime(2014, 5, 14, 15, 24, 54, 234333))
```

```
Out[47]: Job(u'569decc8-7ad2-41eb-83cc-353d7386d2b9',

enqueued_at=datetime.datetime(2014, 5, 14, 15, 24, 54, 235954))

Out[47]: Job(u'155c493e-5a2c-4dcf-8d9b-3ae2934bf9e5',

enqueued_at=datetime.datetime(2014, 5, 14, 15, 24, 54, 238030))

#xiaorui.cc
```

这个是 worker.py 打出来的日志：

```
23:24:59 Job OK, result = 5

23:24:59 Result is kept for 500 seconds.

23:24:59

23:24:59 *** Listening on high, default, low...

23:24:59 default: utils.tosleep(5) (e91cfcb8-850b-4da4-8695-13f84a6a0222)

23:25:04 Job OK, result = 5

23:25:04 Result is kept for 500 seconds.

23:25:04

23:25:04 *** Listening on high, default, low...

23:25:04 default: utils.tosleep(5) (cc6c78d4-e3b5-4c22-b027-8c070b6c43db)

23:25:09 Job OK, result = 5

23:25:09 Result is kept for 500 seconds.

23:25:09

23:25:09 *** Listening on high, default, low...

23:25:09 default: utils.tosleep(5) (569decc8-7ad2-41eb-83cc-353d7386d2b9)

23:25:14 Job OK, result = 5
```

```
23:25:14 Result is kept for 500 seconds.

23:25:14

23:25:14 *** Listening on high, default, low...

23:25:14 default: utils.tosleep(5) (155c493e-5a2c-4dcf-8d9b-3ae2934bf9e5)

23:25:19 Job OK, result = 5

23:25:19 Result is kept for 500 seconds.

23:25:19

23:25:19 *** Listening on high, default, low...
```

这里在看下官方给的例子：

```
from rq import Connection, Queue

from redis import Redis

from somewhere import count_words_at_url

# 创建 redis 的一个连接对象

redis_conn = Redis()

q = Queue(connection=redis_conn)  # 默认是用 redis 的 default 队列名

# 封装任务

job = q.enqueue(count_words_at_url, 'http://xiaorui.cc')

print job.result    # => None

# Now, wait a while, until the worker is finished

time.sleep(2)

print job.result    # => 889
```

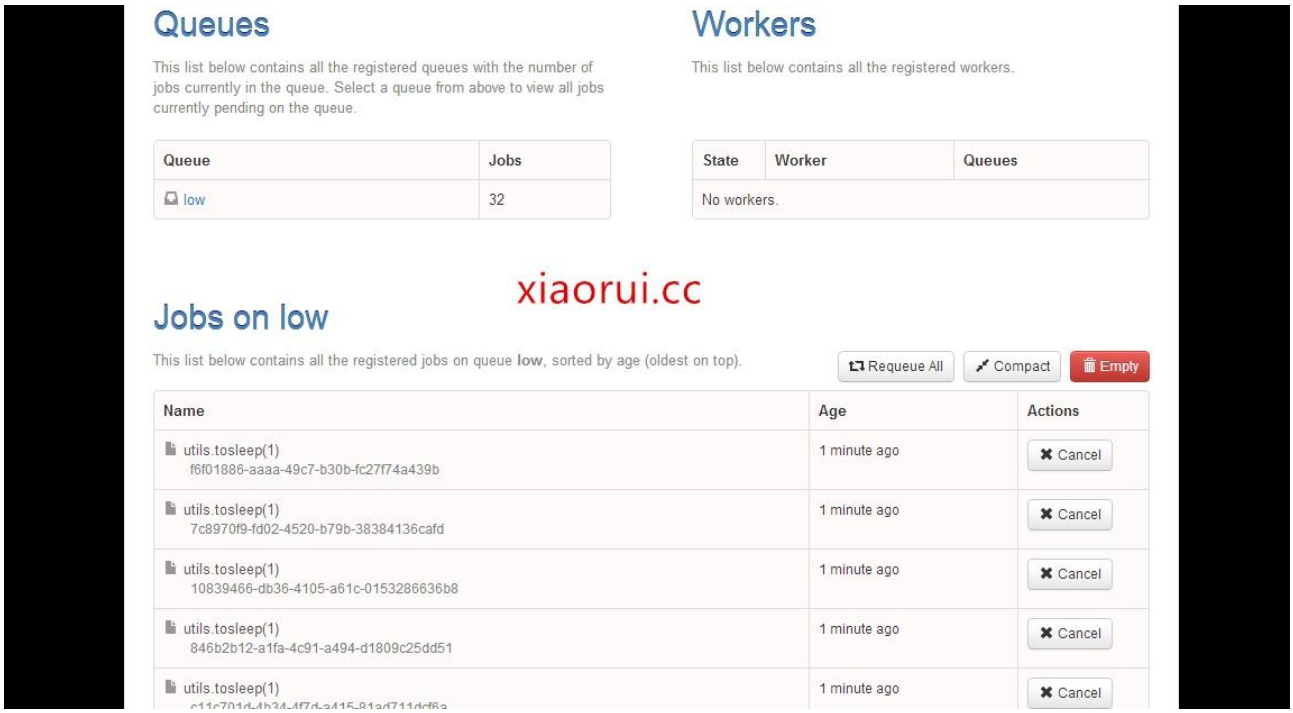
rq 可以设置任务的优先级别的，比如一个 low 级别的。

```
q = Queue('low', connection=redis_conn)

q.enqueue(count_words_at_url, 'http://nvie.com')
```

好了先这么着吧，官方 http://python-rq.org/docs/ 还提供了很多实用的东西，比如装饰器啥的。

对了，官方提供了一个 rq 的管理平台页面。

地址是 https://github.com/nvie/rq-dashboard



Android 开发实践：WIFI 连接功能的封装

作者：tickTick 来源：http://ticktick.blog.51cto.com/823160/1410080

在上一篇文章《Android 开发实践：WIFI 扫描功能的封装》介绍了如何利用 Andriod 的 API 实现 WIFI 的扫描，本文则重点讲述一下如何连接 WIFI 吧，在此，也给出一个封装 WIFI 连接过程的类，提供简单的接口以供在各个代码工程中复用。

与 WIFI 扫描类似，WIFI 的连接同样是一个耗时的过程，所以需要放到线程中执行，通过回调来通知调用者连接结果。该回调接口的定义如下：

```
public interface WifiConnectListener {
public void OnWifiConnectCompleted( boolean isConnected );
}
```

从 Android 的 WIFI Setting 可以看出，一般添加一个新的 WIFI 连接，需要给出三个信息，一个是 WIFI 的 SSID，一个是 WIFI 的密码，另一个是 WIFI 的加密类型，不同的加密方式，连接时程序中的配置是不同的，这里定义一个枚举，给出四种常见的加密类型：

```
public enum SecurityMode {
OPEN, WEP, WPA, WPA2
}
```

Android 的 WIFI 连接过程，总体上分为三步，第一步，添加网络配置，第二步，根据网络配置连接 WIFI，第三步，监听系统的 WIFI 连接状态消息。下面就直接给出示例代码，关键的地方都在代码中注释了。

```
package com.example.testwifi;

import java.util.List;

import java.util.concurrent.TimeUnit;

import java.util.concurrent.locks.Condition;

import java.util.concurrent.locks.Lock;

import java.util.concurrent.locks.ReentrantLock;

import android.content.BroadcastReceiver;

import android.content.Context;

import android.content.Intent;

import android.content.IntentFilter;

import android.net.wifi.SupplicantState;

import android.net.wifi.WifiConfiguration;

import android.net.wifi.WifiInfo;

import android.net.wifi.WifiManager;

public class WifiConnector {

private static final int WIFI_CONNECT_TIMEOUT = 20; //连接 WIFI 的超时时间

private Context mContext;

private WifiManager mWifiManager;

private Lock mLock;

private Condition mCondition;

private WiFiConncetReceiver mWifiConnectReceiver;

private WifiConnectListener mWifiConnectListener;

private boolean mIsConnnected = false;
```

```
private int mNetworkID = -1;

//网络加密模式

public enum SecurityMode {

    OPEN, WEP, WPA, WPA2

}

 //通知连接结果的监听接口

public interface WifiConnectListener {

    public void OnWifiConnectCompleted( boolean isConnected );

}

public WifiConnector( Context context , WifiConnectListener listener ) {

    mContext = context;

    mLock = new ReentrantLock();

    mCondition = mLock.newCondition();

    mWifiManager=(WifiManager)mContext.getSystemService(Context.WIFI_SERVICE);

    mWifiConnectReceiver = new WiFiConncetReceiver();

    mWifiConnectListener = listener;

}

public void connect( final String ssid, final String password, final SecurityMode mode ) {

    new Thread(new Runnable() {

        @Override

        public void run() {

            //如果 WIFI 没有打开，则打开 WIFI

            if( !mWifiManager.isWifiEnabled() ) {
```

```
                mWifiManager.setWifiEnabled(true);
            }
            //注册连接结果监听对象
            mContext.registerReceiver(mWifiConnectReceiver, new
IntentFilter(WifiManager.SUPPLICANT_STATE_CHANGED_ACTION));
            //连接指定 SSID
            if( !onConnect(ssid,password,mode) ) {
                mWifiConnectListener.OnWifiConnectCompleted(false);
            }
            else {
                mWifiConnectListener.OnWifiConnectCompleted(true);
            }
            //删除注册的监听类对象
            mContext.unregisterReceiver(mWifiConnectReceiver);
        }
    }).start();
}
protected boolean onConnect( String ssid, String password, SecurityMode mode ) {
    //添加新的网络配置
    WifiConfiguration cfg = new WifiConfiguration();
    cfg.SSID = "\"" + ssid + "\"";
    if( password !=null && !"".equals(password) ) {
        //这里比较关键，如果是 WEP 加密方式的网络，密码需要放到 cfg.wepKeys[0]里面
```

```
        if( mode == SecurityMode.WEP ) {
            cfg.wepKeys[0]     = "\"" + password + "\"";
            cfg.wepTxKeyIndex = 0;
        }
        else {
            cfg.preSharedKey = "\"" + password + "\"";
        }
    }
    cfg.status = WifiConfiguration.Status.ENABLED;
    //添加网络配置
    mNetworkID = mWifiManager.addNetwork(cfg);
    mLock.lock();
    mIsConnnected = false;
    //连接该网络
    if( !mWifiManager.enableNetwork(mNetworkID , true) ) {
        mLock.unlock();
        return false;
    }
    try {
        //等待连接结果
        mCondition.await(WIFI_CONNECT_TIMEOUT, TimeUnit.SECONDS);
    }
    catch (InterruptedException e) {
```

```
            e.printStackTrace();
        }
        mLock.unlock();
        return mIsConnnected;
    }
    //监听系统的 WIFI 连接消息
    protected class WiFiConncetReceiver extends BroadcastReceiver {
        @Override
        public void onReceive(Context context, Intent intent) {
            if (!WifiManager.SUPPLICANT_STATE_CHANGED_ACTION.equals(intent.getAction())) {
                return;
            }
            mLock.lock();
            WifiInfo info = mWifiManager.getConnectionInfo();
            if ( info.getNetworkId()==mNetworkID && info.getSupplicantState() ==
SupplicantState.COMPLETED ) {
                mIsConnnected = true;
                mCondition.signalAll();
            }
            mLock.unlock();
        }
    }
}
```

与 WIFI 扫描的封装代码类似，这里也用到了 Lock 和 Condition，就是为了阻塞地等待 WIFI 连接的结果，保证正确的 registerReceiver 和 unregisterReceiver 网络连接状态监听对象，同时，设置了 WIFI 连接超时，防止由于 WIFI 模块的问题导致界面收不到回调而长时间“卡死”。

另外，AndroidManifest.xml 文件中记得添加权限支持哦：

```
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"></uses-permission>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"></uses-permission>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"></uses-permission>
```

这个 WIFI 连接类的封装就分享到这里啦，希望对初学者有帮助，java 文件见博文后面的附件，有任何疑问欢迎留言或者来信 lujun.hust@gmail.com 交流。

插件式的 80 后程序员怎样在夹缝中求生存？

作者：沈逸 来源：http://shenyisyn.blog.51cto.com/4968488/1409442

我们先说 80 后。网上到处流传 80 后是最苦逼的、在夹缝中求生存的。

我个人部分同意，为啥说 80 后在夹缝中呢？（这里我们先摒除一些 80 后成功人士，我可以很负责的说，这是少数。如果你就是这少数中的一员，那么请直接跳过本章）

我个人总结了有那么几点：

1. 现在大部分财富都掌握在 60-70 后手中。

这条估计已经是一个不争的事实。话说时事造就英雄，60、70 经历了房地产、下海经商、互联网、软件发展的爆发期，而这个时期为他们积累资金和财富提供了基础。而此时，80 后还在读书。

2. 即将发生的财富转移

话说 90 后甚至 00 后不处于夹缝中，为何？因为 60、70 后积累的财富会在未来 10 年后进行转移。这个财富转移会是一个很平稳的过程，也就是说 60、70 后的财富会直接被 90 后、00 后继承，由他们来继续创造财富和价值。

这里就要讲到 80 后的财富转移。一般 80 后的上一辈都是 40、50 后，尤其是 80 后早期的人。略懂历史都应该知道，除非你的父辈有一定的社会背景，否则 40、50 后的人大部分都正好经历了文革最“苦难期“，所以往往普通的 40、50 群众积累的财富不足以让 80 后产生事业的跳板。

3. 前有古人，后有来者

这里就要讲到“倚老卖老”这个词语。譬如企事业单位，很多领导大都是 60 和 70 后。根据中国的国情，80 后如果要上位必须等待 70 后退休、双规或者意外死亡。就算如此，80 后也已经 50 岁了。

我们再讲到新兴的一些互联网企业。80 后的年龄比较尴尬，在这个年头，80 后正好是在 30-35 岁左右。激情有但是心有余而力不足。很多 IT 企业更愿意去招一些 90 后来补充冲锋力量。因为稳定、发挥价值周期长、猝死的可能性少。

我有个朋友，在类似 BAT 这样的公司干活。本来激情四射，但是当他发现他原本手下的小伙子（90 后）快速达到和他一样的地位时，他退缩了。不是服输，而是他发现人一过 30，精力基本够不上，不足以和 90 后的身体对抗。通俗点，就拿加班来说吧，90 后能通宵两夜不觉累，而 80 后的他加班超过凌晨 3 点第二天就要请假回去睡一天，如果状态不佳第三天上午还来不了。

4. 80 后就是试验品

怎么个试验法？譬如

1. 高考 3+X 是在 80 后身上开始的。记得当年 3+X 高考第一年实行，大规模的低分、超低分涌现。为什么？第一次嘛，都觉得痛，不适应好吗

2. 英语四级和本科学位挂钩。这也是在 80 后读读到大学后开始的。英语固然不能说不重要，但是要和中国大学的学位挂钩简直是千古奇谈。

3. 公务员考试。这点尤其对于 80 后早期很窝火。在这之前公务员都是分配的，并不需要考试。好了，到了我们 80 后能当公务员时，突然要开始考试了，最重要的是啥？关系网开始了，你能不能当公务员和你有没有过硬的关系是有关的。

4. 坑爹的房价。

这么说吧。如果你是 78 哪怕 79 年的，当你能够出来买房时，根据你已经工作三年的经验，一定可以凑足首付，因为那时的房价没有现在这么离谱。而当 80 后大学毕业开始打算买房时，他会发现你就是打工不吃不喝十年才能把首付付掉，房奴也是从 80 后这代人具备典型意义的。

而相对于 90 后，则有 60、70 后把财富转移给他们，00 后还需要买房吗？80 后奋斗了一生，房子就是留给 00 后的。当 80 后咽气时终于把房贷还清了，不过这时对不起，享受和你无关。由你的 00 后子女来 enjoy 吧。

好了，讲了这么多苦逼的事实，那么我们 80 后怎么办呢？难道就自暴自弃？

如果你丧失信心纠错了。每个时代都有每个时代的活法，针对 80 后也有一种活法，那就是插件式的生存。

60 后和 70 后往往大部分和新兴互联网技能是脱节的，90 后的技能没有经过“各种挫折的”洗礼还不够成熟。而唯有 80 后：掌握技能、有一定社会经验、跟进社会潮流不落伍、打拼了这么多年有一定的人脉资源。

这句话可以这么理解：60、70 后要生存靠的是背景、资源和时势；90 后、00 后要生存必须考团队的引领（也就是有个 70 后或者 80 后在带领）

而，唯有 80 后能凭着自己的一技之长独立成活。

所以我们会看到很多自主创业的 80 后，他们创业并不是因为有好爸爸、有好的资源，而是因为他有一个很突出且经过挫折洗礼的技能。譬如开水果店、做共享软件、做网店、做动漫、做游戏等等。这些主力军可都是 80 后，他们靠的是自己的双手，更重要的是他们是独立运作的。

我们把 80 后掌握的技能称作为“插件”。也就是我们这些人能够任意在一个环境中施展自己的才能，而不依赖于某个特定的平台。

我前面写过一个文章叫做“私活小记”。程序员做私活这是很普遍的，我看到身边有些 80 后程序员，大部分后来成家立业都是从做私活开始。道理很简单，有些企业发现给这些程序员私下开发的系统和给公司开发并没有什么两样，而且还便宜。为何不选这些个人呢？

而在一个公司里，你会发现 60、70 后不会干私活至少是不会想把私活“发扬光大”，因为他们要依靠这个企业、组织或生存了这么久的平台为他们养老送终。90 后也不会干，因为“还太嫩”，就算干，也不会发展成一定规模。

这里，我们也是从“私活”这个角度来解释插件这个名词。

我有个 80 后程序员朋友，他除了代码还擅长 UI 设计。于是，他平时会帮别人做一些私活，也就是做 UI 设计。当他后来每天接到的 UI 设计单子发展到一定规模时，他毅然辞去了工作成立了自己的设计公司，5 个人的团队。业务很简单，就是专门帮其他企业做 UI 设计，据说收入非常可观。

这种案例有很多。简单说来，就是通过一技之长形成插件式的生活，并逐步把插件做大、做强。

正因为 80 后的有这种特性，所以也决定了 80 后独特的生活方式：叫做即插即用，独立生活，万事不求人”。

故事结束，不长，也希望对大家有些启发。同时也理解插件是干嘛用的。对于 80 后来讲，也就是我并不做一个高大上的平台，而是做一个插件，你离开我就不行，或者至少不够完美。而融入了我这个插件你会更加锦上添花。

面对公有云，ITPro 该何去何从？

作者：翟老猫 来源：http://3387405.blog.51cto.com/3377405/1404238

这暂时不是一篇技术博客，这是一个 IT 从业人员在大潮中的思考和选择；当然希望随后这些思考也会积淀和回馈出必要的技术博文。

如果您不清楚我的工作，不妨简单说明一下，我是一名 ITPro 工作者，24K 纯纯的，甚至是一名 ITPro 的技术推广者；然而现在很多从事 IT 工作的 ITPro 可能都会隐隐的感觉到云计算浪潮的冲击力，疑惑是否当云计算例如微软的 Azure 公有云中的 IAAS 服务甚至是 PAAS 服务被公司订阅，将会取代传统 ITPro 工作呢？这种多少有点“阴谋论”以为的调调正在潜移默化的吞噬 ITPro 接纳公有云的心，抵触情绪油然而生。感叹生不逢时的人大有人在吧？！

真的会是这样吗？怎样才能在公有云来袭的今天找到自己的定位和价值所在呢？

我作为一个技术推广者，也陷入了深深的思考中，下一步自己将何去何从？作为推广微软公有云的我会不会自己革了自己的命呢？何况市场上不止微软一家公有云厂商，还有著名的亚马逊 AWS，Rackspace，国内的阿里云不一而足！可是仔细想来，到底公有云服务替换的是服务？是软件？还是底层的硬件系统呢？

如果微软的 Azure 公有云替换更多的是硬件端，那么作为一个专业的 IT 人员，那么其实很多找硬件厂商解决的事情例如网络配置，存储划分等是不是可以更快更容易的完成呢？的确一些 IT 的基础性工作在公有云中是交由例如微软 Azure 公有云服务的 IT 运维进行维护的，但是在这个有底层运维服务保障，服务层可以设置故障域升级域，数据层有本地复制和地域容灾复制的基础之上，是不是还可以把精力放到如何快速实现和部署，如何监控优化资源上呢，要知道即便是公有云服务中 IT 的自动化，IT 的层级化监控，扩展仍然是一个不能规避的话题；除非你单独开发一套纯粹符合云特性的中间层服务，那么对于

传统的 IT 企业来说，公有云打破的只是 on-promise 的樊篱，IT 基础设施其实变得更加灵活在其上其实可以演变出更多的套路，从混合云结构的布局，到应用到公有云迁移，从数据备份到容灾场景的扩展；其实公有云提供给了 ITPro 更多的应用空间；对于一个初创企业也许在创业初期开发者和 ITPro 可以没有清晰的界定，一个人可以身兼数职；但是一个成熟的企业，面对若干成熟的 IT 应用，需求和业务场景，怎能通过一个简单的云计算定义转换就褫夺了 IT 从业人员的宝贵经验？更何况在混合云场景，更需要了解企业 IT 结构，了解迁移方式，了解适用于 IT 应用场景解决方案的人员呢？

所以，更快的了解公有云服务，帮助公司实现通过公有云降低企业 IT 投资，与此同时参与契合点的寻找工作，实现真正符合公有云，私有云贯穿的云操作系统；这样的系统开发者喜欢，ITPro 也会找到自己的位置，当你利用混合云敏捷性实现了快速的虚拟机从私有云向公有云扩展的同时，确保了配置一致性，安全性，那么你的价值其实相对于过去是在提升而不是消逝。

云计算的确是一场不折不扣的革命，但是从农业革命工业革命到信息化革命这样多的过程，有些行业的确面临着挑战，但是这些挑战不会因为你的对立就此消失，恰恰相反，这些挑战推动了一些力量，一些让这个世界变得更好的力量！我深信 IT 行业瞬息万变，但是即便在云计算风起云涌的几天，把握机遇迎难而上，定位准确仍然可以凸显我们的价值。

两年手游创业失败的总结

作者：毕成功 来源：http://passover.blog.51cto.com/2431658/1404115

最新的一款游戏发布上线，表现虽然比以往产品要好，但是还是没达到预期。虽然感觉已经找到原本可以维持发展的模式，但迫于目前的资金压力和时间成本，团队士气已经非常低落，即便有一万个不情愿，但也不得不放下了。创业过程中虽然经历了很多困难，但坚持两年也实属不易，而最后还是以失败告终。纵观整个历程，有诸多感慨，写下此篇祭奠这两年的奋斗，再重开自己一段新的历程。

【创业的准备条件】

这应该是我失败的根源，自己没有任何手游的背景，就直接拉拢几个技术人员一起创业。其实最重要的就是团队，这个我是意识到了，但是没有意识到的是团队的组成。只有技术人员的团队大多数创业都是失败的，因为创业不是只有开发产品，还有开发什么样的产品和怎么把产品卖掉，也就是说必须要有一个产品经理和销售经理，这才能算是一个完整的团队。当然，你有其他合作方履行这样的职能也是可以的。

如果再给我一次机会的话，我一定会先去一个知名的手游公司做上一年，然后再拉出来一个产品经理一起做，销售渠道方面必须要有先建立起自己的关系网。这样规划好一个产品，能做出来，再卖掉就更有机会实现盈利。或者自己的资历还不够再出来创业，就在一个各方面能力健全的团队里面去做。

【选什么创业方向】

我所做的事就是一个 CP（内容提供商），做了几个产品才发现原来 CP 是整个生态链的最底端，当然也是最容易进入的。这是一个最苦逼的事情，要不断的进行开发和维护，如果产品赚钱当然有很不错的盈利，但是如果不赚钱基本就默默的死掉了。而上游的代理商和渠道商则高端很多，他们不用开发具

体的产品，只要做平台就可以享受所有应用的分成，而渠道慢慢拿到的利润比 CP 还要高，但是渠道商可没那么容易做，建立一个渠道的成本也是非常高。

但是我之所以能撑这么久，其实算是选对了方向，因为做游戏开发的资金回笼很快，一直有持续的资金流。当然如果做渠道的，那就是要抄概念，拉到资金继续发展。

所以如果你能做一个渠道的事情，那就别考虑做一个内容提供商的事情。尽量做资金流回笼快的事情，或者做概念能拿到投资的事情。

【发展的步伐】

如果梦想着创业就能发财那就失败一半了，我觉得更重要的是看到风险并找到对策，我就是对风险评估的太过乐观。这个应该要分两种情况：一是是否已经具备所有相关条件，这种应该就是在所在行业内有足够积累再跳出来自己做；另一种是还有所欠缺的情况下出来做，这就一定要有风险预案，没一个靠谱的解决办法就是一个巨大的隐患。

发展的步伐也要根据以上情况做不同应对。如果风险大，那就别先想那么远，更应该先从一些简单快速的事情上手，先让团队运作起来看到希望，也能维持现金流。如果万事具备，那当然可以规划的更大一些，先接手一些投资，然后做一款具有竞争力的产品，再推出去。

例如自己进入了一个全新的行业，上来第一款作品就花了近一年时间开发并维护了一个难度较高网游，本以为只要做出来就有足够盈利，其实完全是没有根据的幻想。其实我应该上来先用 3 个月时间开发一个简单的单机游戏，摸清开发的流程，找到相关的人脉，而且单机游戏即便不维护也有持续性收入，不至于像第一款产品那样现在几乎绝收。而且在缺少产品策划的情况下，应该先从成熟产品的模仿甚至是接外包来起步，等时机成熟再去做更大收益的产品。

【创业的收获】

我从来不认为只有成功才有收获，这两年我全心全意的投入，除了觉得对不起和我一起做的兄弟，没有太多遗憾。但在这个过程中，还是有很多收获的喜悦，简单总结一下：

1. 管理上，我明白了只是管理好团队还差很远，我原以为稳定的团队、高水平的技术、良好的组织沟通、以及团队氛围这些就是已经把团队管理好了，但我现在认识到这只是表面，根本是让成员有成就感并获得相应回报，失败的项目就不会有成功的团队。

2. 技术上，我完成做了一个新的技术领域，从技术选型、系统架构、功能开发等都做了一遍，是一个非常挑战性的过程，但是我还是很好的做出来了，获得了很宝贵的经验。另外还用了多个云平台和 Nosql 的技术，做了自动化测试框架等一些比较主流的技术，有了新的技术积累。另外在美术短缺的时候自己做了主美的工作，对美术规划能力有了突破性提升，自己也能弄一些美术了，很神奇的一个经历。

3. 产品上，认识到自己不能盲目自信，虽然自己是游戏资深玩家，但是并不能代表愿意付费的广大玩家，这还是要有足够的数据才能说明问题，还是应该在一些大公司学习别人积累的经验才是最快的成长方式。

4. 销售上，这是原本自己从未直接涉猎的领域，现在知道和代理商、渠道商是个非常利益化的关系，双方都有好处才会有销售机会；也明白了用户价值是根本，只有用户价值超过推广成本才有正向的利益推动，才能做大。最后基本了解了手游这个生态圈的运作模式。

5. 公司运作上，原来上班都是做着相关的事，公司运作层面的也没怎么了解过，自己从公司创建一直到运行两年，现在已经有了清晰的认识，虽然不是什么难事，但是能亲身经历一次的人并不多。

我的人生不是只有两年，但我用我最宝贵的两年买来了一次无价的人生经历，这将受用终身，相信这段经历会为我未来的发展铺垫了坚定的基石。

程序员在囧途之垃圾创业团队

作者：沈逸 来源：http://shenyisyn.blog.51cto.com/4968488/1402707

以前"**空虚和寂寞**"时写的一篇通过真实案例进行”小说化改编“文，原型中的”我“不是作者本人。特此拿出和大家分享，也与自己共勉。

正文：

这年头互联网创业有两个人就算一个团队了，如果是精英组成的团队往往两个人能抵得上十个人，但如果是一帮平庸之辈呢？那么你会发现你的团队简直是一个澡堂大杂烩。

半个月前有着九年软件从业经历的我还在一家公司任技术副总监，公司的各项技术和项目被我整理的井井有条，年前还被公司吸纳为公司小股东。我的管理模式其实很简单，就是粗暴式管理，严格考核项目人员，只要月底总体考核和客户评价连续三次低于 80 分，那么我会无情的让人力部门通知该员工可以卷铺盖走人或者换部门，反正我这不留“次品”。

这个管理模式是我上任后认为采取的最有效的办法，就拿技术人员来说吧，较早先公司内部考核机制比较松散，员工积极性不高甚至出现了上班打游戏的现状，我的授业恩师也是我的提拔推荐人--------公司的元老外加技术总监胡总看我是个管理苗子，特地向老板推荐让我成为公司的技术副总监，据说公司成立到现在第一次设立技术总监的副职。

恩师临上任前给我的任务是：把技术团队整好，必要时可以淘汰部分员工。

有了授业恩师的支持，我开始了大刀阔斧的改革和淘汰“不合格”员工，并由我亲自主持编制了新的技术人员考核制度，从现在看来确实有点暴，不过我认为如果不暴根本无法压制不住当时的不正之风。

改革中也遇到了很大阻力，第一波考核后有一个项目经理和近七位开发人员被我纳入了辞退名单，项目经理带着若干人等气势汹汹的到恩师那告状，说我是因为以前还是愣头兵的时候批评过我而公报私仇给他们小鞋穿，恩师思考了一会儿，大手一挥：“这个事既然交给他负责，那么一切以新的规定为准，以后不要来找我”。

恩师的”无原则”支持让我感动的差点泪涕同时喷出，这就是信任的力量。

两个月后，改革终于见到了成效，员工工作效率得到了大幅度提高，甚至连他们平时在公司走廊里走路的步伐都加快了很多，我很满意。与此同时，恩师再次大放光彩，在全体员工会议中声称虽然我是副职，但是以后项目上一切以我的决策为最高标准。

于是，员工终于认可了我实际的技术老大地位，我身边的跟班也多了，开会时我的决策不再有人反驳，有部分项目经理听到我某些明显有瑕疵的决策后，除了拍手叫好还能立刻当场感同身受的举一反三，还有部分项目经理当场表示我的技术决策能力比胡总更高一筹，我默默的表示这话我爱听。

我终于体会到权力让你飘在天空中是一种什么感觉，我认为这是一种吸食鸦片都无法带来的垂直愉悦感，尽管我从来没有吸食过鸦片，而且这种愉悦感很快让我在工作中忘记了恩师的存在。

不过没多久我的“嚣张跋扈”引来了“杀身之祸”。

二、恩师出山

最近我发现已经很少召集大家开会的恩师突然出山了。这件事源自于公司新接了一个大型开发项目，当我最终决策使用 java 来开发这个项目时，手下几个核心项目经理偷偷告诉我恩师已经召集他们私下开过会了，要求这次项目使用 c#来开发，并且重新启用我早先已经抛弃的 mysql 数据库。

我在那仅比恩师小两个平方的办公室发呆了半小时，明显感觉到恩师的做法并不是以技术为出发点的。

“什么时候开的会？”我问身旁的号称最忠于我的项目经理张伟。

“昨天晚上，QQ 上接到的通知，我以为是你召集的，结果开会时发现只有胡总在。老大，你和胡总不会有什么事吧？”张伟煞有介事的问我。

自尊心不允许我承认，其实我心里已经隐约感觉到这件事不会这么简单。

果然不出所料，恩师中午突然通知所有技术人员召开项目大会，主题是项目人员工作分配，会议上恩师告诉大家我最近工作很辛苦、很劳累、很憔悴，希望大家更加努力的拿出主观能动性为领导分忧。

整个会议我似懂非懂的跟着恩师的节奏，很快我手下的 80%的人员被划拨到这个新建项目，恩师还表示他将亲自负责本次项目的建设，目的是为我排忧解难，好让我充分发挥我的技术优势，把更多的精力放到公司核心技术的攻关上，不要在“没有技术含量的”管理事务上浪费太多时间。

剩下的没有被划拨的 20%的人员都是我上任后亲自招募和培养的新人，看着恩师似笑非笑的脸颊，我突然明白了什么。

这天下午，恩师的办公室门关闭了半天，门外围了一圈窥探真相的人。原因是我和恩师发生了严重的争吵，恩师第一次在我面前露出了不和谐的表情，大声训斥我自从上任副总监后在管理上完全没有建树，员工技术能力也没有提高，而且很多项目经理在背后投诉我，最后恩师还表示这次他重新出山完全是为了我好，希望我回去“一点点排查我的缺点”。

我从吃惊转为激动最后转为气愤，其实我都明白，我从最初恩师手中手感颇佳的指挥棒已经变成极其令人讨厌的绊脚石，我唯一没有排查到的缺点就是我没有激流勇退。

(三) 拉山头

经过了三天的冥思苦想，我认为：与其被恩师当做傀儡般的摆弄不如自立山头。

人往往在一努之下会作出连自己都不敢相信的决定，一周后我把辞职报告拍在了恩师的办公桌上。

恩师没有阻拦，甚至没有任何表示，只是随后吩咐人事部督促我离职时的办公用品归还，并委派了一位他的亲信项目经理监督我的代码移交，并立即更改了公司技术服务器的密码。

我终于体会到什么叫“人还没走，茶就被倒掉了“的滋味。

临走时，我私下召集我那 20%的亲信，表示我打算自立山头创业，有愿意跟我走的我负责他的后半辈子，不愿意的也不勉强，以后还是朋友。

今非昔比，20%的亲信瞬间有 17%因为家庭原因、身体状况、自身能力状况等反正是我以前从来听闻过的原因婉拒了我。连以前号称结婚都要经过我同意的项目经理张伟也表示最近家里经济压力大、家中老母身体也不是很好，如果排除这些原因他是很愿意跟我走的。

我笑了笑，不是我不想戳穿他，而是戳穿了意义不大。

三天后我正式离开了公司，同时跟我一起辞职的有四人：早先我还是普通开发人员时共事过一段时间的大黄，两个刚被我招进公司还没转正的吴谨和薛明，还有个是我也不知道怎么会跟我一起辞职的美工孙翠。

大黄：在我眼里技术能力很一般，当年和我在一个项目组共事过一段时间。随后我升了项目组长、项目主管、项目经理、部门经理直至最后的技术副总监，而大黄依然是开发人员，只不过头衔前加了个“高级”二字，要不是实在没人，否则他就是跟我辞职我也不会要。

吴谨和薛明：大学应届毕业生，技术能力一般。当时是因为一个项目急需 java 开发人员，所以被我招进来打打下手，本想项目结束后直接给点安家费让他们另谋生路，不过这次事情一来，他们估计永远不会知道我当时有这个想法了。

孙翠：我几乎从来没和她说过话，我查了查员工表，她应该设计部新招的美工，虽然已经转正了，但是由于前段时间我太醉心于“管理上的权力斗争”，所以根本顾不上认识这些新来的员工。

辞职当晚，我请看起来不怎么样的四位尾随者吃了一顿大餐，祝贺我们的新团队组建，也提前祝贺一大波美好未来将向我们涌来。

（四）垃圾团队

九年的工作经验让我积累了一些技术资源，并有着自己开发的一套相对成熟工作流软件也是公司的主打产品之一，虽然临走时恩师“胡汉山”派了两个人盯着我移交笔记本和电子文档，但是技术这个东西主要是思路和设计理念，它们都在脑子里就算赤裸裸的给你看也看不懂。我花了近两个星期修改了一些原来懒得改的 bug，重整了一些性能低下的框架设计，并略微调整了 UI 界面，我认为凭着这款软件。

繁琐的公司注册流程结束后，我宣布正式进入创业阶段。

前期人手少但是分工还是比较明确的，其中我着重培训了吴谨和薛明，让他们按照我的架构设计继续根据客户的需求开发新的模块；孙翠负责用户体验方面的完善，使其更加人性化；而我负责客户的挖掘和市场的开拓。

基于以前的客户资源我连续接到了两笔单子，由于都需要给客户演示再加上分身无术，我只能亲自负责一家规模较大的单位，而让吴谨和薛明负责另外一家公司以前的一位老客户单位演示。

貌似顺畅的人员分工很快出现了问题，也许是我太高看了这两位新人，当我颇为顺利的完成了第一家单位的演示并在合同上盖章时接到了另外一家单位的电话，该单位以前和我有点交情负责人告诉我：就我们现在这种产品功能，是不可能让他们考虑选择我们的产品的。

我一惊，难道我们演示的版本不一致？

回答公司后问其缘由，原来是演示环节出了问题，在演示前客户提出要定制一个流程给他们演示，由于实在无法两头兼顾我把这项工作交给了吴谨和薛明负责，于是这个环节还真出了问题，演示过程中新模块的 Bug 像洪水般的涌出，甚至连最基本的流程自定义功能都没走通，而且孙翠配合他们做的功能页面也被指责为“像坨屎”。

我火冒三丈，薛明竟然当场顶口说是去之前在公司里程序已经全部调通了。

对于这种程序员不肯承认错误的通病，我再次当场暴了粗口，并勒令晚上哪怕通宵也要把问题改好，搞不定第二天全部给我走人。

第一次见我发这么大火，孙翠在旁边默默的红了眼圈。

“不许哭，以前在公司出了差错也许可以容忍你们到月底，现在出了差错我们就得喝西北风”我把火气顺便也撒到了孙翠身上。

“我们其实已经很努力了”孙翠委屈的解释。

“那只能说明你们是垃圾了”，我丢下这句话摔门而出。

晚上和一帮狐朋狗友聚会，朋友问起我现在手下有几个精英，我乘着酒劲又暴了粗口：“精英个屁，垃圾团队”。

朋友脸色吓青。

(五) 开除员工

这件事过去后不久，我开始加强对这些队友的培训，虽然被我断定为“垃圾”但是有总比没有好。我请来了我以前的几个技术专家好友轮番对他们进行技能培训和创业思想传导，见吴谨虽然资质一般但是还算刻苦并且对技术无比热爱，于是我亲自手把手的把公司产品连架构到代码注释通通给他详解了一

番，并把一些我认为不该开放的模块源码全部交给了他，我的目的只有一个：我要培养出一个技术接班人。

我给孙翠报了当地的一个 UI 设计提高培训班，那里面有专业的 UI 设计大牛驻站教课，我听了一堂试讲课后认为：除非你是傻子要不绝对不会学到完一点治疗效果都没有。

我见薛明虽然技术稍差但性格开朗，由于其本人也有意愿于是便培养他做销售方面的工作。平时除了带他去用户现场实战外，也给了他报了当地一个高级销售培训班，我听了一堂试讲课后也认为：除非你是白痴要不绝对不会学到完变成傻子。

结果薛明首先让我失望了。

外地项目的一次竞争性谈判，正好我也在出差，由于项目难度不高而且我已经提前给用户方打过了招呼所以就放心的交给了薛明去投标，结果薛明在投标现场还是搞砸了。该标还没开就因为标书某些细节没有按照文件要求编写相同格式结果被废标。

回来后经过详细了解情况我才知道，那天招标时某个评委被另外一家供应商打过招呼了，由于怕被我们中标所以该评委特意在评审标书阶段楞就是“拿着放大镜般”找到了一个规定表格格式不同的瑕疵，直接性把我们提出了局。经“审问”标书是薛明主笔，吴谨和孙翠辅助。

薛明竟然连这种低级的错误都会犯，简直是枉费我对他一片信任。

同时我看着两次即将到嘴的肥肉都被这几个“垃圾”搞砸了， 再也忍不住了，除了吴谨和孙翠连带被我暴粗口外还问候了薛明的某位家人。

那天，终于也忍不住爆发的薛明和我在办公室动起手来，薛明的衣服被我扯破，我那柔顺乌黑的头发被拉下来了一撮。事后，我当场宣布开除薛明，薛明喊了句“谁爱干谁干”后摔门而出。

片刻后吴谨表示他也想走人，我想都没想就同意了，随后我告诉孙翠如她也要走我绝对不留，这丫头呆呆的看着我半宿没吭声。

凭啥我几个也在创业的朋友手下的队友如此犀利而我的队友如此平庸，他们遇到的队友不是技术大牛就是销售大兄，就算原来不是经过培养各个都修正了正果，难道注定我就是个失败者吗？

俗话说不怕神一样的对手就怕猪一样的队友，我觉得这句话还要扩展一下：不怕猪一样的队友就怕怎么配种都生不出小猪的队友“。

(六) 金子和垃圾只有一步之遥

薛明和吴谨走后只剩下孙翠一个人，这小丫头最终还是选择继续留下，她告诉我：她在公司时就特别崇拜我，只是希望我以后能平和一点、心态好一点。

我第一次流露出感动的表情，不过很快被我的假装镇定掩盖了。

由于公司小品牌度低招人成了一个老大难问题，成熟的 IT 人员不愿意屈就，新出炉的大学生我又怕重蹈吴谨和薛明的覆辙，于是高不成低就导致我连续一个月都没有找到合适的员工。无奈之下我只能自己甩开膀子承担了技术、销售、售后等所有岗位，至于孙翠，我认为她只能老老实实的干好美工活，超出她能力以外的事情我一律采取不信任状态。

俗话说是金子一定会发光，是垃圾永远只能呆在垃圾桶。但随后一件事让我发现原来有时垃圾和金子就差一步之遥。

冤家路窄，半年后在用户召开的一次软件项目供应商标前讨论会上我遇到了老东家，恩师“胡汉山”也看到了我，这厮随即向我发送了蔑视的眼神。我明白：“报仇的机会”来了。只不过要想报此仇难度系数不比人类登月低多少。

据我了解到，我以前的亲信部下张伟是这次项目的技术经理。张伟跟了我很久，了解我的一切技术套路和我手上的产品技术核心;同样我的功力也出自恩师之手，他也了解我的一切老底。这就意味着如果用常规的产品技术层面去对拼明显无法打动评委，如果打价格战更加没希望，因为这次投标是公开招标，技术先进性和需求一致性是最大的打分项，如果刨除一切因素评委肯定更加倾向于在当地有着“常青树”之称的老东家，除非我在产品上有创新点和突破点，可短时间内如何能这么容易的实现呢？

距离正式招标还有一个月，我推掉了所有小项目和杂事，专心在家研究产品方案和谈判技巧，无奈“知己知彼”是我这次和老东家竞争的最大问题，因为我们双方太了解了，在技术层次差不了多少的情况下，恩师团队的力量就明显占了优势，如果说我们双方同时想到了一个创新点，那么恩师只需拉开架势让团队成员开足马力加班半月绝对能整出来，而我就算一个月日夜无休都不可能赶得上他。

孙翠这几天也魂不守舍的整天不知道电脑面前干些啥，我也懒得理她，爱干啥干啥去，如果这个项目失了标的，我认为以后也没脸在圈内混了。

临近正式投标的日子越来越近，我始终没有想到好的亮点去打动评委，就在我抓狂之际，孙翠突然要求我去她电脑面前看个创意。

我漫不经心的走到她电脑前，顿时眼前一亮。

“这是什么？”我看到了一个用户体验极佳的产品界面，而且 logo 还是我们自己的产品标识。

“老大，看你最近郁闷没好意思打扰你，就拉上吴谨、薛明对产品做了重新的 UI 设计和功能改版，你看看好吗？”孙翠一脸骄傲。

这里我谈一下我和恩师共有的一款产品。这是一款专门用于建立企事业门户的 CMS 软件，由恩师早期带领团队开发了第一版本，我接手后沿用其框架开发了第二版本，我另立山头后我通过更换界面和修改部分功能以及无耻的贴上自己公司标签后变成了第三版本，不过我知道恩师在我走后也组织力量开发

了第三版本。其实不管是恩师的版本还是我的版本都有一个共同的缺陷，那就是使用者素质要求太高。一般性的企事业技术型用户需要培训后才能上手，如果业务型客户要自行建立一个模块或者子站那几乎是不可能的，虽然我们产品中无处不在的宣称“任何人都能快速建站”，实际上如果没有一定的 html 基础知识根本无法做出一个合格外观和功能的站点。

我仔细查看孙翠手上的版本，发现其实内核和功能都没有做变动，唯一有变化的只是在软件中加了三个细小的功能：

第一个是向导。原先的后台需要通过点击不同的菜单栏配置多项数据后才能搭建出一个站点原型，并且整个操作流程中充满了各种技术术语，如果真没有技术基础的人是不可能上手使用。而现在吴谨和薛明在这个基础上架设了一个向导界面，用户在向导界面上一路“下一步”便可完成整个站点原型的搭建，并且各种技术术语全部做了人性化修改，譬如“选择页面模板”改成了“选择界面的样子”，“建立数据源”改成了“创建页面中的新闻来源”。虽然更改后在技术人员眼里看起来很可笑，但是从用户角度来看却是有很大的亲切感。

我顺便查看了一下吴谨和薛明的代码，写的依然没有技术含量，只不过是每一步向导都调用我核心模块的一些接口，然而恰恰就是这些“没有含量”的代码产生了一个新的用户体验。

第二个是模板下载功能。大家知道市面也有很多共享 CMS 软件，这些软件都有自己的社区和免费或者收费的模板，当我们需要使用不同模板时需要去这些专有社区中下载。然而，卖给企事业单位的 CMS 完全不是这样的，一旦软件提交给用户后服务到此结束，如果用户有新增模板的需求则要重新谈判项目建设费用，这就是导致很多企事业单位对现在的一些做 CMS 软件的公司痛恨很深，因为一旦他们完成了第一版本网站建设后就无法更换模板，因为模板都是统一提供的，要想更换模板就得加钱。

吴谨和薛明充分结合了共享 CMS 软件的功能，在软件中加上了模板更换功能，并且可以整站预览。每个模板免费使用半年到一年不等，如果继续使用则要收费，这对于企事业单位来讲完全够了，同

时对于后期我们和用户另签合同谈模板收费也是一种无形的导向性。在 IT 界要想从用户那边忽悠到钱，那首先不能让用户对付钱这个事情感到反感。

我也顺带查看了代码，一看就是薛明的幼稚风格，只是加了一个模板列表页面，当用户选中模板后调用我的替换模板接口进行预览，然后取消后再替换回来，从技术层次这简直就是借我的布料做嫁衣。

第三个是右击菜单插件。其实网站管理员在查看网站时如果想临时新增一个新闻，他就得先另开一个新浏览器窗口然后登入后台，多次验证后才能进入新闻发布界面，然后填写内容，点保存然后审核然后再发布到网上。这导致以前也有很多用户觉得太麻烦，有的用户提出譬如他在某网站上看到好的新闻想直接发布不想打开后台。对于这种我当时认为“无理”的要求我采取了直接性拒绝或者报了一个高价改造的方案。

而吴谨和薛明用的办法很简单，做了一个注册表文件，点击后你的右键菜单自动会加上“发布新闻”这一项，你只要选中随便那家网页的文字右击并选择发布新闻，则这条新闻的发布已经自动的完成了。

我再次浏览了吴谨和薛明的实现代码，依然很无节操，他们只是新增了一个页面，然后调用了我内核中各种发布新闻的接口，当外部页面 post 数据过来后自动执行发布。

最后，是孙翠的 UI 设计。其实也是很没有节操的一个改动，即把页面中所有出现的表格、按钮和一切带边框的元素改成了圆角，并在通用页面底部加上一段脚本，使鼠标滑动表格或者按钮时产生渐变色差，最终导致产生一定的立体感。

我惊讶的看了看这几个变化，不能说这几个改动具有天才的创新程度而且很多理念都是从互联网上抄袭过来的，但是这些变动可以说让整个软件变的更加有血有肉和人性化，可以说在某个领域平凡的几个功能加到了不同领域的产品中随即产生了不平凡的变化。

（七）写在最后

招标的结果果然大暴冷门，我竟然中标了，评委当时的评语是：“好用、有创意，软件充分考虑到了用户的感受”。

恩师“胡汉山”临走时向我发射了“嫉妒”的目光。

数日后我把已经在其他公司上班的吴谨和薛明高薪请了回来，这两个人竟然告诉我“他们早就等待着这一天了”。

那晚我真诚的请我的“垃圾团队”们把了一次酒言了一次欢。

后来我有一次问孙翠她为什么崇拜我。

她红着脸告诉我，“因为我不够优秀，跟着我做事没有压力”。

如何找到自己的个人价值？

作者：双鱼座小龙 来源：http://liuqunying.blog.51cto.com/3984207/1403513

毕业至今两年有余，之前从没考虑过这个问题，总是认为把领导交给我的事情做好了就一切都 OK 了，就算做不好也能找到各种理由或者说借口吧,为自己的结果做出推脱。

2014 年年初，经历人生的第一个坎，与自己的初恋女友分手，感觉人生一片灰暗，感受到现实的残酷。回看自己的所作所为，只能用没有资本，不够成熟来概括。

工作上也经历了很多事情，看着以往关系不错的朋友一个个离职，有些还是我认为很优秀很敬佩的人因为工作没有得到老板认可而离开去了其他公司的。从那一刻，我突然感觉到了危机感，感觉工作瞬间没了光亮，说不定哪天就因为工作能力或者因为工作没有得到领导的认可而被炒鱿鱼。

当人有了危机感就会焦虑，但是光焦虑是没有用的，深刻认识自己以及弄清楚自己以后的路要怎么走才是最重要的，也就是职业发展规划的问题。

穷则思变，这句话绝对没错。

我是谁？我想要成为怎样的人？我的价值在哪里?

当你感觉自己前途不太明朗的时候，你就得问自己你有哪些能力是别人认可以及能够给你的工作带来优势的。

自己的价值只能自己去发现，别人是没法告诉你的。老板只会关心，活有没有人干，不会关心谁去干，更不会关心干这件事的人将来应该是如何的，当然也有比较有管理智慧的老板，这个就不在我们个人要考虑的范围了。毕竟，我们不能期待每一个老板都是为员工发展的好老板吧。

很多人都说三年是一个坎，无论是恋爱还是工作，三年都是一个很奇妙的时间，到了这个时间点都需要重新审视自己。过去的三年自己有哪些改变，未来的三年要变成什么样子以及要怎么努力完成目标？

无论是程序猿还是运维狗，过去三年你是忙着复制代码还是重启服务了，未来的三年你还想继续复制代码和重启服务吗？我相信任何一个人都不愿再花三年时间把原来的事情再重复做下去，哪怕你能在复制代码的时候能顿悟或者在重启服务的过程中升华，是时候问问自己我的价值仅限于此吗？

过去三年做了太多杂事，由于公司业务开发较多，运维人员过少（可能老板并不这么认为），每天的时间基本都被各种开发修改配置文件，重启服务，问题诊断，扩容机器，梳理监控等事情占据，很少有时间能够静下心来看书思考，全身心投入谈一场不分手的恋爱。也许，这也是运维岗位的共同问题吧，所以做运维的同学如果你有幸找到妹子愿意与你共度余生，那还是学会珍惜，无论如何都别放手，如果你没找到的话，先闭目沉思，自我反省三分钟，然后继续去修服务器吧。言归正传，也许因为工作的各种问题让我们无暇顾及自身的发展，碰到比较极端的情况才会突然惊醒，作为一个闷声不吭，埋头苦干的屌丝如何不被老板炒鱿鱼，不被女神鄙视。

跟公司里面工作经验比较丰富，级别比较高，思维还非常清晰，干劲还十分足的同事一比，我们的战斗力瞬间就降到最低了。看到每次领导把比较重要的活都交给这些人去做的时候，你会不会有一种人的差距怎么大，而且怎么都赶不上的悲凉。说实话，就算人家每天不这么努力，对于像我这样大部分人来说也是很难望其项背的，更何况人家比你更加意识到努力的重要性。

我们就真到 low 到如此不堪的地步吗？

前段时间，我参加了一个有很多公司的 IT 人员参加的交流会，在问答环节，很多人提出的问题都是很低级的，让人从心底感觉到提问者专业能力的缺乏。很多问题搜索引擎里面搜索下关键词都能找到非常专业详细的解答，但是很多人不知道是懒得去查还是就理解能力问题导致搞不清楚。如果你刚好也如

我此时的感受，会不会心里顿然有种莫名的优越感。好吧，这些简单的问题还有人不懂，我还是懂的，看来我还是可以的么。

不要妄自菲薄，不要心生轻狂。在一个公司做到优秀不是真的优秀，到任何公司都能做到优秀那才是真的优秀。

这句话是我去参加一个公司的面试后得到的结果，面试的过程就不说了，结果是不尽如人意，很多概念自己都没搞清楚，但是这些概念一看就又都很容易弄明白。

人都是存在自己的长处和短板的，是弥补自己的短板，还是充分发挥所长。

每个人都是有自己的价值的，关键是如何找出自己的个人价值，然后努力去提高自己的价值，这样你就不用担心哪天被老板炒，因为你清楚自己的价值就清楚在哪里能发挥自己的价值。

如何找到自己的价值，个人认为可以通过以下方式，当然欢迎大家提供自己的方式：

1. 交流经验

交流不限于同行交流，跨行业也可以交流，不限于跟自己的同事交流也可以跟自己的客户交流。很多时候，你的客户可能更容易发现你身上的闪光点。

2. 深入研究原理

浮于表面肯定是无法有大发展的，要想有长远的发展还是需要深入研究自己所接触的东西。搞清楚为什么，很多时候能快速提高自己的能力，也能增加自己的价值。

3. 面试展示与交流

面试，不仅仅是我们找工作的一个步骤，更多的时候我们可以为了准备面试梳理自己所掌握的东西，在面试的过程也是发现自身问题的过程。所以面试失败，不代表我们就此失败，更多的是一个帮助我们成长的环节。通过面试我会记录别人问我了什么问题，我怎么回答，有哪些东西是我没弄清楚或者跟别人无法讲清楚的，其实讲不清楚归根结底还是自己没有弄清楚。

4. 结伴交流与成长

本来我想说带徒弟的，但是后来想想这个说法不如用结伴成长来说的确切。

带徒弟可能更多强调了一个能力强的人带能力弱的人成长，比较偏向于单方面的带领作用。结伴成长，大家一起成长，一起发现问题，一起解决。每个人思考的角度不一样，得出的结论一定会有差异，消化差异就是一个成长的过程。

5. 看书，学习

看书吧，虽然说网上东西很多，但是也很杂，容易让人精力分散。

以前，前女友要求我每周读一本书，我认为这是不可能的，但是我现在发现我错了，每周读一本书，只要挤挤时间或者换份工作还是可以做到的。

很多时候不是做不到，而是不想做，只是给自己找借口而已。

看书能让人有系统化的认知成长，当然看书不限于技术书籍，看看人文书，也有利于个人成长。

6. 跨界发展

你的价值仅限于做一个 IT 男么？

换个环境或者换个圈子，说不定你的价值就会显现出来。

有人就是不喜欢写代码，有人就是不喜欢看各种 IT 书籍，可以考虑看看其他方面的东西，说不定你就爱上她，一发不可收拾。

百度实习面经验-JAVA 研发方向

作者：可以_先生 来源：http://paradisewj.blog.51cto.com/7435653/1402217

收到百度上研实习的 offer 差不多有两个星期了，前段时间挺忙，这次算是抽点时间总结一下三次电面，顺便给大家当面经用吧~

从投简历到第一次约面试差不多有两个星期吧，三月中投的，等了半个月都没消息，以为没戏了，清明回到家准备爽一番，结果正在 dota 中，先阿里来了个电话，后面没半个小时百度又来，当时一点没心情，怕弄砸了便约了隔几天一面。(阿里说后面会再来电话，结果毛线消息没有，魂淡 - -!)

闲话不多说，一面如期而至，说实在话，第一次面试挺紧张的，所以在约好的时间提前到学校的楼下散步去了，希望环境能帮助自己调节一下~电话比预期的晚了半个小时，一上来简单的介绍后便进入正题，结果一上来就问我算法(心中我了个大擦啊，劳资算法最屎啊，这是要跪的节奏啊)，第一个问了单链表如何检测有环，其实这题在之前看面经的时候好像还真瞄过。。可惜当时貌似没看答案，在面试官的提示下说出来了，第二个问的是两个有序数组的归并，这倒是真心简单，也没什么难点，我当时犯浑，说插入，其实意思倒是对的，就是说法有问题吧，好在面试官准备说了一点的时候，我后面接上了，解释了一下之前我的表达有误，第三个算法可能有点复杂吧，关于二叉树的，他说让我记下，我说在外面，不在实验室，不方便记，结果居然就过了。

不问我算法了！！！(各种窃喜的节奏)后面估计就是大家各种 JAVA 面试的常菜了，String,StringBuffer,StringBuilder 的区别，应用场景，HashMap 和 HashTable 的区别应该准备过面试的都能答出来的，说 HashMap 和 HashTable 的区别的时候我说到线程同步的问题，他又开始问对于线程同步的理解，如何让两个线程交替输出 a,b，后面的话记得不是特别清楚了，应该问了一些项目方面的，感觉答的挺顺利的，还有让我写个 sql，答错了 - -。一面差不多就这样了吧，当时结束感觉虽然算法和 sql 都答的不好，但总体关于基础方面都答的还行吧，主要的原则就是问我一个点，如果我知道里面

原理的话，我就尽量把原理也说出来，比如内存的分配啊什么的，如果问的不知道的话，也尽量把自己的分析说出来，总之就是尽量的"秀"吧。其实面试官有时候不一定要你正确的答案，而看你的思路是不是正确。(一面 40 分钟)

二面等了将近一个星期，本来以为跪了的节奏了。倒是万神各种怒砍 offer，周围的人也开始各种讨论面试，心中自然着急的节奏。二面当时约的要在电脑旁边，说电面过程中会发一些题目过来，需要编码，我靠，传说中的远程桌面么。

实际证明就是邮件发题目过来而已。不同一面，二面一上来就问项目，换了个角度而已，问我中间遇到的挑战啊，如何解决啊，系统怎么调优的，插一段，说来之前有投过一个阿里的内推，虽然没有正式一面，但是当时学长先给了个电话问了我的情况，当时请教了学长关于项目的准备，他就重点提到了这些问题的准备，这里感谢一下~这些问题准备了的话，答得其实还是挺顺利的。然后面试官问我有没有写过 servlet，虽然那是我最开始学 JAVA 的时候写的，基本没写多少，但此时怎能示弱~写过！他让我写个用户登录的 servlet 发给他，可能怕我写的慢吧，他发来一个，让我检查有什么问题，说来也巧，之前看线程安全的问题的时候，关于 struts2 的线程安全的问题让我困惑了，就各种 google 线程安全问题，看到说 servlet 线程不安全，所以一眼看出来那题的 point 了。

然后让我写 jdbc 的连接，同样他发了一个 demo 过来，让我找问题，sql 注入的问题不用多说，记得看微博的时候陈皓转过一个关于汽车故意将车牌写成一个语句的，说能让监控产生漏洞的，说来虽然是玩笑，道理是这么说通的。面试官虽然用的是 PreparedStatement，参数却还是用拼的方式连接起来的。然后的问题是各种 try,catch，finally 的处理都没有，这个其实当时我只记得大概各种连接，结果集的关闭什么的要做异常处理的，具体的没有很清楚，但是很庆幸，我可以百度！秒度之，翻到最下，拷拷改改发去搞定~哦，忘了，中间还让我写了个标准的单例模式，这其实也在陈皓的酷壳上看过不止一次，面试前还特地在 workspace 中写过一个，写了个相对来说，考虑到线程安全的版本发给他，结果谁知道他居然要求更高，还是找了一点问题，其实也是，还是有可能出问题的，交流一番，可能觉得我表

述的不好吧，让我重新写了一个给他，双重锁的，建议大家准备单例模式的话，可以看看陈皓那篇博客(福利：http://blog.csdn.net/haoel/article/details/4028232)。

最后两个问题让我挂了电话半个小时后发给他，一个是智力题，不难，几分钟搞定，另外一个是两个数组归并(没错，又是他...)要求是要一个性能最好的方法，当时 20 分钟的时间，我哪有时间想啊...写了个最普通的方法发过去了，备注了句"时间紧迫，未做其他思考.."，感觉好窘迫 - - ！ 二面给我的感觉就是面试官比较注重细节，很多大家平时容易忽略的问题被拿出来问了，说运气好可能就是这部分吧，觉得这段时间看的一些资料啊博客啊之类的没有白看。(二面一个小时)

三面很快，二面当天的晚上就约了隔天三面，挺开心的，这么快约三面说明希望大嘛？但也挺紧张，直接导致当天晚上各种睡不好。三面的问题比较零散，给我的感觉更注重解决问题的能力吧，直接问些知识的并不多，让我介绍项目的时候，很多的时候面试官会中断一下，然后设计一个场景，问如何去实现等等，比较综合。也会问一些关于技术学习，目标，长久规划的问题，这些大家有点想法的相信都不会是问题的。最后一个问题是道设计题，差不多是网上的博客的公共评论系统的设计，这里不赘述了，时间到的时候提了个想法上去，感觉还算不错吧。三面时间很长，一个半小时，不过当时也没有太深感觉，很多都是问题都是聊出来的，对于三面大家感觉不用可以准备什么，自己好好发挥，展现出来平时水平就 Ok 了。

下午收到了北京的电话，说愿意给 offer，不过后来因为入职时间的问题，调到上海去了....

最后，我也只是只小菜鸟，相信大家的 offer 都在来的路上，祝大家好运~

扫描二维码，加 51CTO 博客为朋友

关注 51CTO 博客微信，打造你的个性！

http://blog.51cto.com

编后语

《51CTO 博客月刊》为 51CTO 博客整理出品

如果您有意投稿，请联系我们
如果您有反馈意见，请告诉我们

联系方式：
Email：blog@51cto.com
QQ: 1173854158

欢迎关注我们：
@51CTO 技术博客